



The Graphics Workshop Graphics Primitives for Microsoft Compiled Basic

This manual was written by Brian C. Gledt, with contributions by Ethan Winer and Don Malin. This manual was designed and typeset by Jacki W. Pagliaro. Entire contents Copyright © 1989–1994 by Brian C. Giedt, Ethan Winer and Crescent Software.

Na portion of this software or manual may be duplicated in any manner without the written permission of Grescent Software, Inc.

CRESCENT SOFTWARE, INC.
11 ANLEY WARTUNE
RIDGEFELD, CONNECTICUT 06877-4505
(203) 438-5300
FAX. (203) 431-4626
Offescent's Support BBS: (203) 426-5958
Second Printing 11/94)



LICENSE AGREEMENT

Crescent Software grants a license to use the enclosed software and printed documentation to the original purchaser. Copies may be made for back-up prohibited, and adherence to this requirement is the sole responsibility of Copies made for any other purpose are expressly However, the purchaser does retain the right to sell or distribute programs that contain Graphics Workshop routines, so long as the primary purpose of the included routines is to augment the software being sold or distributed. Source code and libraries for any component of the Graphics Workshop program may not be distributed under any circumstances. This license may be transferred to a third party only if all existing copies of the software and documentation are also transferred.

WARRANTY INFORMATION

Crescent Software warrants that this product will perform as advertised. In the event that it does not meet the terms of this warranty, and only in paid, if notified within 30 days of purchase. Proof of purchase must be meet the advertised claims.

CRESCENT SOFTWARE'S LIABILITY IS LIMITED TO THE PURCHASE PRICE. Under no circumstances shall Crescent Software or the authors of this product be liable for any incidental or consequential damages, nor for any damages in excess of the original purchase price.



Table of Contents



	1-1	1-1	1-2	1-3	1-4	1-4	1-5	1-6	1-8	1-9	Multi-Tasking Manne	transport transport to the second transport transport to the second transport tran
											-	÷
										:	:	
			•									
		•		•								
		•	•	•	٠							
	•	•	•	٠	•							
	•	•	•	•	•	٠	٠	•				
		•	•	•	•	•	•	•	•	•		
			•	•	•	•	•	•	•	•		
			•	•	•	•	•	•	•	•		
			•	·	•	•	•	•	•	•	•	•
			•	•	•	•	•	•	•	•		
						÷	•	•	•	•	٠	•
				÷		•	•	•	•	•	•	•
								•	•	•	•	•
								•	•	•	•	•
_								•	•	•	•	•
_	•									•	•	•
	•								i.	:	•	•
_	•										•	•
	•	٠,	۵.									
_	•	٠ ;	5		•							:
	•	٠ ج	<u> </u>	•	•							
	•	. 5	2	•	•	•		S				
ō	•	. ;		•	•	•	٠.	5		٠.		
	•		=	•	•	•	. ,	ĕ		. 2	Ξ.	
\overline{c}	·	-		•	•	•	•	~		. :	2	
ΙŠ	i.	. 6	3		•	•	٠ ر	2		. (2	
-5	·	: :	≦ ક	3	•	•	٠,	1)	•	٠,	<i>;</i> ,	0
Õ		. 7	<u> ₹</u> ∂	5	•	•		<u>ਤ</u> ਹ	0	ج ٠	₹ :	⊒ .
2		. 9	2 .	3		0	٠	Ξ ;	5	٠ ي	۶ ۲	<u>.</u>
		. 6	5 6	3		\$	• 3	J 7	3	٠,	≥ 5	7
		. `	. 3		. 4	, ,	0	5 6	4	2 د	₹ 7	٦.
		٠ 4	5		. :	: 4	3 6	5 8	3 7	₹ ?	5 8	4
	Ξ.	. '5	٠,٠	۲	. 0	3 6	5 C	۽ ر	= c	íδ	5 :	7
_	₽;	<u>ج</u> ج	_	. 6	3	(5	. +	<u>ئ</u> م ہے	-		, -2	5
<u></u>	: t	2 5	9 5	5 0	5 -≟	-		7	7	3 9	ے ر	đ
Chapter 1 - Introduction	Introduction	Starting with Granking Wallahan	Naming Conventions	Functions	Multiple Screens	Craen Modes	Different Conditions of the Co	Passing Parameters, Systems	Standard Code	Why use COMMON Cure	3 5	
÷ ·	0	DE		2	=	d	ب د	7	100	2 >	7 :	
0	7	2 5	2		3 3	1 -	. J	1 0	3 6	عے ہ	: :	3
a	= (V.	Z	ĮI.	. 2	. 0	5	2 6	. 0	3	: 5	
-	-									_	-	1
0												

Chapter 2 - Demonstration Programs

S S S S S S S S S S S S S S S S S S S	
	: :
	: :
	: :
	: :
Harrier Control of the Control of th	
P & . & . &	
4	200
HE WELL BELL BELL BELL BELL BELL BELL BEL	BAS
- 5×× E C S F F S E F S B C Y B C S E B B B S E B B B S E B B B S E B B B B	
# Z O Z K D C C C C Z Z K E Z P C C I Z I	2 2
#EBBCCCCEEEEEEEEEEE	20
# X X X X X X X X X X X X X X X X X X X	SS
Demonstration Programs ABOUTPCX.BAS DEMOBOX.BAS DEMOBOX.BAS DEMOCURC.BAS DEMOCURC.BAS DEMOCURC.BAS DEMOCURC.BAS DEMOCURC.BAS DEMOCURC.BAS DEMOCORD.BAS DEMOCORD.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCONT.BAS DEMOCURC.BAS DEMOCONT.BAS	DEMOSCRL. BAS
D A O O O O O O O O O O O O O O O O O O	田田
	D

Assembly Routines

Republication of the common of

Section 2:

...

CRESCENT SOFTWARE, INC.

Section 3: Screen12 Screen13

Screenii

General Screen Man ClearVE DrawByteVE ClearScreenArray

GetLastVCoords. GetLastXCoord % GetCharacter% DrawByteVEOpts
Fail-2EGA
Fail-EGA

GetScreen Mode %

OMOVELLE OMOVESARE OMOVESA

CHESCERT SUFFMARE INC

.3-34	3-37 3-38 3-40 3-41 3-41	.3-45 .3-46 .3-47	.3-50 .3-51 .3-52 .3-53	3-56 3-58 3-58 3-60 3-61
56			VE	
GPrint0C2 GPrint0H GPrint0V25	GPrint1VE . GPrint2VE . GPrint3V256 GPrint3VE .	LineBFVE LineBVE LineStepVE LineToVE	LineToStep Creen	en3 en7 en9 en11 .
5 5 5 5	5	ëëëë	LineToS LineVE Screen0 Screen1 Screen2	Screen3 Screen7 Screen9 Screen11 Screen12

Section 3:

3-62	2 63		40-5	50-5	70-5	60-6		5-/3	3-74	3-75	3 76	0/-5	//-0		3-81	3-83	3-85
		:	:	:	:	:	:	:	:	:		:	:		:	:	:
	:	: :	:	:	:	:	:	:	:	:			:	:	:	:	
		:	:	:	:	:	:	:	:	:			:	:	:	:	:
es.			· ·					:	:	:				:	:	:	:
Routir				•					:	:			: :	:	:	:	:
lation	:						:	:	:	:				:	:	:	:
fanipu	J.										%			:	:		:
een M	enArra	:	VE .	VEOp	• :		ter%	oorde	7 100	oord%	Mode 9	: ш	[II]	1 m	: :	:	:
General Screen Manipulation Routines	ClearScreenArray	ClearVE	DrawByteVE	Draw Byte VEOnts	Fade2EGA	FadeEGA	GetCharacter%	GetLastXCoord %	V 400	Jet Last I Coord %	GetScreenMode%	GMove1VE	GMove2VF	SMono2VE	1 0 0 0 0 1	GSorolly E	OII V E
Gener	Cle	Clea	Dra	Dra	Fade	Fade	GetC	GetI	Catl	סמנד	GetS	GMC	GMo	CMC	CIVIC	Const	1360

Display Box File Display Box File Display PCX File Displa

Interlude1
Interlude2
LineBAS
LoadFont
LoadOutlineFont

HandlePCXP. | tte Get Width % . . . GPaintBox . . . GetOutlineWidth% GetMouseCount% GEditor DrawText · · · · · FullZoom · · · · DrawCursor ...

GerCountLocation

. ■

CRESCENT SOFTWARE, INC.

ORESERVATION WASE INC

Liskend Liskend August Liskend On Night and On Night and

layBox	gue4 gue5 gue6 4-13 gue6 3DButton 4-15 Cursor 4-17 Fext 4-17	00r 4-19 untLocation% 4-20 ouseCount% 4-21 utlineWidth% 4-22 ttlineWidth% 4-23 BOX 9-24 BOX 9-24 BOX 9-25	neFont G	all
DisplayBox DisplayBox Elli DisplayPoxFile DisplayPoxFile DoSeguel DoSeguel DoSeguel	DoSegue4 DoSegue5 DoSegue6 Draw3DButton DrawCursor DrawText Fill7com	$1 \cup 1 \cup 1 \cup 2 \cup $	Interlude1 Interlude2 LineBAS LoadFont LoadOutlineFont LtsMenuG LtsZMenuG MsgBoxG	NightFall OutlineText PCXCAP PositionBox PullDownG PullDnMSG RandomFade SetGPFont SetGPFont SetGWSpacing SetGWSpacing SetGWSpacing

3-106 3-109 3-109 3-110 3-111 3-115 3-115

Appendix B:

The Palette Palette How The EGA Sto

What Can You Use

Appendix C:

Step Lext 4-54 VertMenuG 4-54 WhichPCXScreen 4-55 WhichPCXScreen 4-57 Other Files on the Disk 4-58 Chapter 5: QuickSegue 5-1 About The Script Language 5-1 Letal About The Script Language 5-1 INTERLUDE number "anystring" 5-2 LOAD "filename" buffer 5-3 SEGUE seguetype subtype color delay 5-3 SEND buffer 5-3 SEGUE seguetype subtype color delay 5-3 Segue Types 5-3 Segue Types 6-2 Using the Vector Font Editor 6-2 Using the Vector Font Edit menu 6-3 Items on the Help Menu 6-6 Using Vector Fonts With Your Program 6-6 APPENDICES 6-7 APPENDICES 6-8 APPENDICES 6-1	izontal
---	---------

Appendix E:

Combining with Gr

Converting from Q Converting From (

Improving Pixel
GetCuthe/
ReDrawVE
ResetCathe

Appendix D:

Detailed Listing Of The GPDat% Arr Description of the Blements Used By

CHESTERN SOFTWARE INC.



Introduction

Chapter 1: Introduction

Introduction ORCHANDAM WAS WO Among the deficiency of the control Overview (pictBASIC in creating gi have been to copy popular dia lay graphics may see a didining are adaptives that the did not that there are a did not that the did not a SUR. The bank with a All of the progra show how they ar The Graphics We for noday's inter-for noday's inter-designed for use designed for use 3. This mai advanced 2. BASIC s perform write you Assembl ment ove have sign

INTRODUCTION

for today's interactive, graphics-based applications. These routines are designed for use with Microsoft QuickBASIC 4.x and BASIC 7. There The Graphics Workshop contains the low level graphics routines needed are three key components to Graphics Workshop:

- Assembly language routines that provide a dramatic improvement over what is possible by using BASIC alone. Some of the routines improve on BASIC's speed and code size, while others have significant capabilities not available in BASIC.
 - BASIC subprograms and functions that serve as examples and perform a variety of chores that would be tedious or difficult to write yourself.
- This manual, which provides insights on some of the more advanced topics in graphics programming.

All of the programs include heavily commented source code, not only to show how they are used, but also to explain how they work.

Overview

QuickBASIC has always been a simple and easy-to-use language for creating graphics applications. Built into QuickBASIC from the beginning have been the capabilities to create simple graphics on most of the currently popular display types. It has the ability to draw lines, circles, manipulate While these basic doesn't have the ability to display pre-created images, such as PCX files created in PC PaintBrush. QuickBASIC doesn't give you the ability to QuickBASIC OuickBASIC doesn't have the ability to draw lines using one of the arithmetic operations such as XOR. The XOR technique operates by retaining the old information on the screen so that a successive XOR operation will replace the abilities are adequate for simple graphics applications, scroll a region of the graphics screen in any direction. graphics images and display text in graphics mode. original image on the screen.

Among the deficiencies found in QuickBASIC for graphics applications are the following: Although QuickBASIC can print text in the graphics modes, the process is slow and a programmer cannot specify the backprovides are also slow, particularly reading and writing a single pixel. In ground color of the text. Some of the other graphics routines QuickBASIC terms of code size, if you have ever tried to create an .EXE version of a program using QuickBASIC graphics routines, you will find that it turns into a very large program.

speed of getting a pixel's color off the screen. tion to find the pixel's location. This provides a dramatic increase in the memory to get the color of the pixel on the screen or even use multiplicaroutine uses a simple cache so that it does not always have to go to video DrawPointVE. POINT and PSET, statements. These statements are GetPointVE% and also gives you equivalent but faster routines to replace QuickBASIC times the speed of QuickBASIC's PRINT statement. Graphics Workshop to the screen in any foreground and background color, and at up to 10 or simply a straight replacement (the BASIC LINE statement allows only statement and also has the ability to specify XOR, OR, AND, operators straight replacement). Workshop gives you a line routine that is faster than QuickBASIC's LINE scroll any region of the screen in any direction you choose. and VGA graphics modes. Graphics Workshop gives you a routine to has the ability to display PCX files created from Hercules, CGA, EGA, supplies you with the capabilities QuickBASIC lacks. Graphics Workshop Should all these limitations make you think twice about using QuickBASIC for your graphics programs? The answer is "No". Graphics Workshop In addition, the Graphics Workshop's GetCacheVE% Graphics Workshop gives you a way to print strings

Different Cook

This section talks

enough memory

This section talks

Passing Parame

with which not ex Workshop, and ex Multiple Scree

used with which Naming Conve

This section will

advisable to read routines in a pt After eximining demonstration of easy to alentify

screen modes and manipulate pages, you may be able to fulfill all of you coming out with much smaller executable programs graphics needs without using QuickBASIC's SCREEN statement, thereby Finally, because Graphics Workshop provides low level routines to set

Starting with Graphics Workshop

effectively use Graphics Workshop. In addition to providing a listing of each routine and its calling syntax, many other details are described in This manual covers the many important topics you will need to know to

■ The GPDat/(4) a

Standard

started right away, simply start QuickBASIC 4 with the GW.QLB Quick However, if you are familiar with BASIC programming and want to get Library like this:

QB /L GW.QLB

as follows: If you are using BASIC 7, start it and specify the GW7.QLB Quick Library

QBX /L GW7.QLB

demonstration programs start with the letters DEMO, which makes them quickly see what the various routines do, and how they are called. you may run any of the Graphics Workshop demonstration programs to Once your version of BASIC has been started, with the appropriate library

ON BENINGE MANAGE

Naming Convention

from one system to

This section talks at (Appendix D)

Converting from

routines to your ext

If you are already QuickPak Professio

values can be utiliz routines. It expla This section goes

Dur Ranko sign ur O Destr Mouseauge

All of the easy to identify from the QuickBASIC editor Files menu. demonstration programs are documented in Chapter 2. After examining the demo programs, the next step would be to use the Before using the routines, advisable to read a few of the important sections of this manual. your own. routines in a program of

Corne Grantis ice BASICS LINE R. AND, operators

tenent allows only Hay to print strings n, and at up to 10

St. Wichia

Naming Conventions (Chapter 1)

This section will help you to understand which routines are meant to be used with which screen modes.

Multiple Screens (Chapter 1)

GetPointVE% and S GetCacheVE's ave to go to video

en use muliplica. tic increase in the

ice OuickBASIC's

This section talks about video memory and how most video adapters have enough memory for two complete graphics screens.

Different Coordinate Systems (Chapter 1)

This section talks about the different coordinate systems used by Graphics Workshop, and explains a coordinate system used by Graphics Workshop with which not everyone is familiar.

Passing Parameters (Chapter 1)

I routines to set fulfill all of you

This section explains how most of the routines in Graphics Workshop pass It explains that including the file GWDECL.BAS will alleviate any problems. information.

Standard Code (Chapter 1)

This section goes over the standard code which should be added to your programs to use the different routines in Graphics Workshop.

The GPDat%() array (Appendix C)

need to know to ling a listing of

This section goes over an array that is used to share information between It explains the different elements of the array and how their values can be utilized. routines.

If you are already a Crescent Software customer and are using either QuickPak Professional or GraphPak and want to add Graphics Workshop routines to your existing programs, we suggest you read:

and want to get

WOLB Quick

Converting from QuickPak or GraphPak (Appendix D)

This section talks about the differences and what is required in converting from one system to another.

Ouick Library

Naming Conventions

In this manual and in all of the routines there is a standard naming convention used to specify the screen mode with which the routine is to

called. Most h makes them OFTHARE, BIC

ANT DECESSION TO STATES OUT AS THE

a to printing

considered backs

256-color mode have the letters V256 at the end of the routine names end of the routine name. which work for only the EGA screen modes have the letters EGA at the is used for the VGA and EGA high-resolution screen modes. The routines letters VE at the end of the routine name. This specifies that the routine GMoveIVE and GPrint0VE. You'll notice that both routines have the high-resolution screen modes. be used. A majority of the routines work for both the VGA and EGA Those routines which work only with VGA These routines have names like

Functions

execute statements not related to calculating the result of the function. earlier versions of the BASIC compiler is that user-defined functions can An interesting capability of QuickBASIC 4 and BASIC 7 not present in

> sufficient memor In all modes on t a parameter of th

little-known feat screen. The act visible page and a

Statements work pround and then

pages. Early VG. tion graphics mod

the variable when the routine finished routine could return information was to pass it a variable, and then examine written in assembly language. In the past, the only way an assembler Beginning with QuickBASIC 4 and BASCOM 6, functions may also be

which exceeds the

That's 80 x 480 x anough memory

Dianos notalio

denominator. Thi

QBHERC.COM is resident in memory: the HercThere% function which checks to see if the MSHERC.COM or is used extensively in Graphics Workshop in those cases where returning Now, however, assembly routines may return a value directly. This feature functions must be declared before they can be used. For example, consider value is appropriate. It is important to understand that assembler

IF NOT HercThere% THEN PRINT "MSHERC not loaded"

Screen Modes

mode. The forma QuickBASIC uses

know that HercThere% isn't simply an integer variable. If the function hasn't first been declared, BASIC would have no way to

GWDECL.BAS does not increase the size of your compiled program. Graphics Workshop are in the file GWDECL.BAS. All of the DECLARE statements you will need for routines contained in Including

Multiple Screens

page is assigned a distinct screen number starting with page 0. complete individual screen is considered a page of video memory and each ability to store more than one complete graphics screen in memory. Each mapped and object graphics operations. Some graphics modes have the Graphics mode is an optional mode that supports the use of both bitnormal mode of operation for these cards and supports only text output. All IBM-compatible graphics hardware, the Hercules, CGA, EGA, and VGA display cards, have both text and graphics modes. Text mode is the

CHESCENT SUFFMARE INC

S. Carrier A Section of the Sect

T (A) WH (A)

e name names.

C 7 not present in ined functions can

of the function.

tions may also be Hay an assembler and then examine

ctly. This feature Where returning that assembler

performed on any of the video pages. QuickBASIC maintains both a visible page and an active page. The visible page is the currently displayed screen. The active page is the page with which all the BASIC graphics considered background screens as they are not visible at this time. It is not necessary to draw on the visible page; drawing operations can be Any other pages can be Using this simple but little-known feature of QuickBASIC, screens can be drawn in the background and then displayed instantaneously. The active page can be set as statements work whenever they are executed. starts out as the visible or foreground screen. a parameter of the BASIC SCREEN statement.

In all modes on the Hercules, CGA and EGA graphics adapters, there is sufficient memory to store at least two graphics screens. VGA high-resolution graphics modes, however, don't have enough memory for two graphics Early VGA adapters had only 256K of memory, which simply isn't One high resolution screen requires 640 x 480 x 4 bits to store the information. That's $80 \times 480 \times 4$ bytes, or 132K. Two screens would then require 264K, ly, in the PC world programmers are required to work for a least common denominator. This means that even though the hardware designers fixed their mistake and later released VGA cards with more memory, there is a significant number of VGA cards which simply are not capable of two which exceeds the memory a VGA adapter has to work with. Unfortunateenough memory to store two of its high resolution screens. video pages.

Screen Modes

QuickBASIC uses the SCREEN statement to set the appropriate graphics The format for the screen statement is as follows: mode.

SCREEN Mode, ColorSwitch, ActivePage, VisualPage

Including

ge O. Page O

nodes have the

ly text output.

JA, EGA, and ext mode is the e of both bitnemony. Each

Introduction

10 479. Charts on an VGA graph

graphics mode sc point appears at

The Mode value can be selected from the following table:

All mafar	13	12	11	9	7	10	∞	2		_	-	0	MODE
750 04 0000	VGA	VGA	VGA	EGA	EGA	EGA	EGA	CGA	le	CGA	CGA	Color	MODE MONITOR
All references to 256 000 colors and a :- 41:	320x200	640×480	640×480	640x350	640x350	640×200	320×200	640x200	with a ColorSwitch of 1)	320×200	320×200	Text (80x25)	SCREEN SIZE
	256	16	2	16	2	16	16	2	h of 1)	4	4		COLORS
1 11	256,000	256,000	256,000	2	2	2	2	16	1 300	1 cot	2 sets		PALETTE

262,144 different colors which are available. All references to 256,000 colors made in this manual actually refer to

Hercules Graphics Mode is not a supported standard in the IBM PC BIOS program. A Mode value of 3 is used when setting the Hercules graphics QuickBASIC provides support using a TSR (terminate and stay resident)

To talk about

to use the text mo mean that Grap variables like Row mode coordinate S

going to define th graphics - we are To make it simple time, to manipula approximately 2, screen. In addi mode there is con create less efficies graphics modes r If you've tried w 0) as the base con

has 640x480x4 bi manipulate. A

program modes, including a Hercules screen statement that does not require a TSR Graphics Workshop has a replacement routine for each of these screen

page, and then after you have drawn everything in the invisible backcomplex screens instantaneously. ground, change the visible page. This gives the appearance of drawing When programming you can set the visual page to other than the active

The answer comes mix coordinate syste a row, On a text so

of memory. If we c horizontally on the could work with th

time. The organiza To change every pr The graphics syr

Different Coordinate Systems

screens. These screens have 80 horizontal coordinates and 25 vertical 1 and 80, or 1 and 25 respectively. coordinates. The most common coordinate system is that used by conventional text When you specify a coordinate, you use a number between

screen has a resolution of 640 by 480 pixels. When specifying a coordinate Graphics screens have many more coordinate possibilities.

CHECKET SOFTWARE INC

method to preserve could have made it p right, though you ca you will notice that you create a DOS wi course speed. For memory. The advar coordingte systems,

on an VGA graphics screen, the values range from 0 to 639, and from 0 to 479. Charts and graphs usually start from some center point. This point appears at 0, 0 on the number line. To keep graphing simple, graphics mode screens use the upper-left corner of the screen (position 0, 0) as the base coordinate.

If you've tried working with both graphics and text, you'll notice that graphics modes run slower. This is to be expected. It's not that people create less efficient code for graphics mode programs. Rather, in graphics mode there is considerably more work done to place information on the In addition, there are simply more bytes of information to A typical character mode screen is an 80x25 matrix or approximately 2,000 (2K) bytes. A typical high resolution VGA screen It takes more CPU power, and thus more time, to manipulate 132K as opposed to 2K. has 640x480x4 bits, or 132K. manipulate.

To make it simple to understand which type of coordinate system - text or graphics - we are dealing with in the Graphics Workshop routines, we are to use the text mode coordinate system while running in graphics mode. To talk about a graphics coordinate system, we will use variables like mode coordinate systems we will call the variables Row % and Col %. Any This doesn't mean that Graphics Workshop uses text mode, but it does have the ability going to define the variable names that refer to each system. variables like Row1% and Cols% also refer to this system. XPos%, Y1%, XPixels%, or YPixels%.

actually refer to

256,000

e IBM PC BIOS. nd stay resident) of require a TSR

To change every pixel on the screen would take an exorbitant amount of time. The organization of EGA memory stores 8 pixels in every location of memory. If we could generalize what we want to do with the pixels we could work with these 8 pixels at a time. The 8 pixels are all in a row horizontally on the screen. This allows for 80 groups of these 8 pixels on This makes it possible to The graphics system of an EGA has 224,000 different pixels on the screen. memory. The advantage in working with a screen 8 pixels at a time is, of The answer comes from understanding how a computer stores graphics course, speed. For an example of a major graphics program which mixes you create a DOS window and place it in a re-sizable window on the screen you will notice that you can't move it just one or two pixels to the left or right, though you can move it any number of lines up or down. Microsoft could have made it possible to position it at any pixel, but they chose this a row. On a text screen there are 80 columns. This makes it possible to mix coordinate systems. You might ask, "Why mix coordinate systems?" coordinate systems, look at either Windows or Presentation Manager. method to preserve speed.

> invisible backnce of drawing

than the active

unher hetween

ONDECL BAS keyword is places routine in Graph

still have a different pixel color anywhere on the screen understand that the 80 column resolution is just for positioning. You can Similarly the VGA high-resolution mode has 480 lines. It is important to horizontal coordinates. use this new system will use variables like Cols% and Lines% to define vantage of this speed as well. This mixed system will use text mode columns and graphics mode lines for its coordinates. The routines which Similarly, Graphics Workshop has many routines designed to take adregion. Graphics modes using this mixed system will have 80 The EGA high-resolution mode has 350 lines

Standard Code

Below is the stance is for the main pr Two torms of stan

Insert other

spot. a Window2D which will be described below. GWDECL.BAS. Each of the coordinate systems has been given a name relating to the types in this file. We will refer to the text mode coordinate system as system 0. Then the graphics mode system will be called system Window1, or Window2. There is also a definition for a Window1D and type which defines a window on the screen. each system. This system will be called system 2. Each of the different coordinate systems has been placed in the file The last system is the mixed system with text columns and pixel lines This is called Coord0, Coord1, or Coord2. One is the type which holds the coordinates for a single There are two TYPE definitions for This is called Window0 The other is a variable

and another. system allows low-level assembly routines to skip the step of calculating upper-left corner to the lower-right corner of the region. This coordinate The width and height can be referred to as being delta values from the absolute coordinates for the upper-left and lower-right corners of the systems. Window2D. The "D" stands for the delta values which it utilizes how many locations on the screen exist between one side of the window corner of the box and the width and height of the box to define a window. There are two ways to describe a window (box) using these coordinate The second window system uses the coordinate of the upper-left The first is probably more familiar as it defines the window with We will refer to these window systems as Window1D and

Passing Parameters

of how this can make a program faster, take a look at the parameters can be preceded by the BYVAL keyword. For an explanation is for that reason that many of them pass parameters using the BYVAL keyword in QuickBASIC. When a DECLARE statement is made, certain DEMOBNCH.BAS program which demonstrates the speed differences. The routines in Graphics Workshop were designed with speed in mind. It

attempt to call a prothe Graphics Works

modules. This file that you should incl As a safeguard, we that state that i the DEFINT for the are stored and pass expect integer valu The first line is ver

variable jou want to

parameters which are meant to be passed using the BYVAL keyword. But QuickBASIC doesn't automatically understand that a routine

ON 38 WALLSON MANSON We same on Louison SO IT IS Advisable to the file ONDECL.B

New York A Local Sublice

A inputation S. Yucan 1. 12 as 350 libes

your programs and modules, you will not have any problems. The GWDECL BAS file contains DECLARE statements for every assembly if you include the file GWDECL. BAS at the beginning of EVERY one of routine in Graphics Workshop. Below is an example of how the BYVAL keyword is placed into a DECLARE statement:

DECLARE SUB DrawPointVE (BYVAL XX, BYVAL YX, BYVAL PointColor%)

Standard Code

placed in the file X m de coordinate Il be called system heen given a name

on and pixel lines. PE definitions for nates for a single a WindowID and

these coordinate the window with efine a window.

Two forms of standard code will be covered in this section. The first form is for the main program file. The second form is for all of the modules. Below is the standard code for your main program:

'Load font into the Font\$() array 'Includes standard declarations Graphics Workshop Vector Fonts Specify loading font number 1 Specify loading font number 1 'Makes all variables integers Determines monitors attached 'OutlineFont\$() array 'Load font into the 'GraphPak Fonts '\$INCLUDE: 'GWDECL.BAS' '(Insert other DECLARE statements here) CALL SetGWFont(1)
CALL LoadOutlineFont(FontFile\$) *INCLUDE: 'GETVIDEO.BAS' CALL LoadFont(FontFile\$) 'GPFONT.GW' 'GWFONT. GW' '\$INCLUDE: 'GPFONT.G FontFile\$ = "HELV12" FontFile\$ = "HELV" SetGPFont(1) .\$INCLUDE:

CALL SetVideo

The first line is very important. All of the Graphics Workshop routines expect integer values. The DEFINT statement ensures that all variables are stored and passed in this format. Placing a ! or # symbol after a variable you want to be considered as a floating point number will override the DEFINT for that variable. However, remember that integer math is 'Sets the screen mode always faster than floating point math.

WindowID and

op of calculating e of the window As a safeguard, we have provided a BASIC file named GWDECL.BAS that you should include at the very beginning of all your programs and modules. This file contains appropriate declaration statements for all of the Graphics Workshop routines. If you have included this file and then attempt to call a program incorrectly, QuickBASIC will warn you.

ing the BYYAL

eed in mind.

livik at the

an explanation od differences

the file GWDECL.BAS won't increase the size of your finished program, so it is advisable to include it always. GETVIDEO.BAS determines the Although you don't always have to include all of the above lines of code, monitor you have and dimensions the bare minimum size variables (i.e.

supplied with Graphics Workshop. element per array) that are necessary for using some of the BASIC routines

oplications, the

hi-resolution mode to be set. prior to calling SetVideo. based upon the value in GPDat%(31). The GPDat%() array is described in fuller detail in Appendix C. You can modify the value in this variable SetVideo is a routine which sets the screen mode. The screen mode is set You could, for instance, force an EGA

the lines which call LoadOutlineFont. It's as simple as that. to as Outline) fonts, then don't include the GWFONT.GW file and remove is described below. If you are not going to use the Vector (also referred screen. If you are not going to use either of the font systems, you don't need any of the rest of the above lines except the call to SetVideo which Workshop's Vector fonts which can be made as large as is possible on the GraphPak fonts which are small scale fonts. There are two font systems supplied with Graphics Workshop. One is the The other is Graphics

- I at the locations

EuroGPFontsil = -

character set), set which requires er

the DEMOEURO Graphics Worksh The font files EU most useful portio method of drawin to be the line dra to display charac The Graph Pak and in all of your mo

added. standard code below. There are only three things which should always be to load a module. Any modules loaded in this fashion should utilize the File Load option on the QuickBASIC menu system, it gives you the option Below is the standard code for all of your modules. When you use the

'\$INCLUDE: 'GWDECL.BAS' '(Insert other DECLARE's here)

'\$INCLUDE: 'COMMON.GW

There is no need to use floating point arithmetic when integer arithmetic is much faster. In addition, all of the Graphics Workshop routines expect Once again it is always best to have a DEFINT statement everywhere.

Why use COMM

to have this file included in every module. warn you. It will add not increase the size of your final compiled programs this file and then attempt to call a program incorrectly, QuickBASIC will ments for all of the Graphics Workshop routines. If you have included programs and modules. This file contains appropriate declaration state-GWDECL.BAS should be included at the very beginning of all your

purpose array for storing all sorts of useful variables. The following and the GPDat%() array discussed in Appendix C which is a general variables between routines. COMMON.GW holds BASIC COMMON statements used for sharing The variables shared include font information

CHESCENT SUPPLIESE INC Toutine TextClobal, b

which we've created need to have the varia two, three or even in as a parameter. The would be to have eve to access variables for what you want is a v what names you've t program. This is ben you are using the sair to that subroutine. When you create a c section describes how to use COMMON.GW and why it should be used in all of your modules.

> Mary K described F in the language

orkshop. One is the other is Graphics s is possible on the Systems, You don't ctor (also referred to SetVideo which W file and remove

to display characters in the extended character set. In character mode applications, the most useful portion of the extended character set seems method of drawing lines: the LineVE routine. So, in graphics mode, the most useful portion of the extended character set is its foreign characters. to be the line drawing characters. In graphics mode, we have a better The GraphPak and Graphics Workshop Vector font systems have the ability

The font files EURO.GFN and EURO.QFN work with the GraphPak and Graphics Workshop font systems respectively, and are demonstrated by (which requires enlarging the font arrays enough to contain the extended character set), set the variables EuroGPFonts % and EuroGWFonts % to the DEMOEURO BAS example program. To enable the European fonts -1 at the locations shown below in the standard code:

"\$INCLUDE: 'GETVIDEO.BAS' 'GWDECL . BAS' .\$INCLUDE:

> When you use the ves you the option should always be

that

'Set this before reaching the GPFont.GW 'include file \$INCLUDE: 'GPFonts.GW' EuroGPFonts% = -1

SetGPFont (1)

CALL LoadFont ("EURO")

*INCLUDE: 'GWFonts.GW' EuroGWFonts% = -1

'include file

'Set this before reaching the GWFont.GW

CALL SetGWFont (1)
CALL LoadOutlineFont ("EURO")

Why use COMMON.GW?

neger arithmetic

routines expect

ent everywhere.

ing of all your eclaration stateu have included uickBASIC will apiled programs

to that subroutine. This means that you can use the variable T% even if When you create a SUB in BASIC, any variables you use there are local you are using the same letter T% in some other subroutine or in your main program. This is beneficial in most cases, because you don't have to worry what names you've used elsewhere. But there are times when you want to access variables from other routines within your subroutine. Essentially what you want is a variable name which is global. One way to do this would be to have every routine that needed the variable to have it passed as a parameter. The problem is that if a simple routine was nested inside three or even more subroutine calls, each of those routines would need to have the variable passed. Take, for example, the variable Global % It needs to be passed to the routine TestGlobal, but we start in routine A. which we've created in the example below.

sed for sharing

ont information ch is a general

Introduction

SUB A (Global%)

CALL B(Global%)

SUB B (Global%)

CALL C(Global%)

END SUB

CALL TestGlobal(Global%)

he doing things w

selection. Where

These menus have which call the rol using an advance running in the tex

for a key press, bo

This shadow effe

routines create m

Warkshop & a sy

One of the m

SUB C (Global%)

END SUB

taking place. If all that needed to be passed was a single variable it would how hard it would be as a programmer to get anything accomplished. hard to imagine adding eight parameters to every routine, not to mention global information about the screen mode, it would be burdensome. be acceptable, but if it was eight arrays filled with font information, and As you can see a lot of extra, and unnecessary, passing of information is

contents of the COMMON GW file. If you're already using COMMON is created automatically inside the include file GETVIDEO.BAS, shown in the "Standard Code" section in this chapter. Shown below are the to use either of the font systems within Graphics Workshop, or if you use the GPDat%() array inside your program. Note that the GPDat%() array statements of your own, we suggest you combine the two sets of COMwhich you must place at the top of all of your routines should you decide we've created COMMON.GW. COMMON.GW is a BASIC include file of your program. requires that it sees the same set of COMMON statements in every module MON statements into one include file. is that it must be placed in every module within your program. Also, it Luckily, QuickBASIC has the answer. It's called the COMMON state It allows you to make certain variables global. Its only restriction That would be a great deal of typing, which is why

If Action is

calling program.

G and VertMening The Action variable can be instructed t To accomplish this continue your pro-

the calling sequent

COMMON SHARED GPDat%(), Font\$(), FontWidth%(), FontHeight%()
COMMON SHARED OutLineFont\$(), FontSize%(), OutLineHeight%()
COMMON SHARED Tile\$(), AltTile\$()

3 is how wa will be immediately and Acc and display themsel When Action is sett what selection the L been pressed, the o

recommended you place at the top of every one of your modules. The keyword SHARED should also appear on each line as shown above. Note, the COMMON.GW include file is just part of what we have Standard Code" section for more details.

2 also reses toelf to may have been overw Setting Action to 2

CHECKENT SUFFINARE. INC. 3 when the menu re If the menus are vaile polled to see It a key

Multi-Tasking Menus

One of the more exciting capabilities we have provided in Graphics Workshop is a system of menus that can run in the graphics modes. These routines create menu systems that appear to be running just as if they were running in the text modes. A special shadow feature has been duplicated using an advanced routine which can change many pixel colors at a time. This shadow effect can be turned off by commenting out the two lines which call the routine GPaintBox.

These menus have a multi-tasking feature which allows your program to be doing things while the menu system is waiting for the user to make a selection. Where most menu programs simply sit in an idle loop waiting for a key press, both the pull-down and vertical menu subprograms let you continue your program if you wish to do so.

> le variable it would t information, and burdensome, It's ne, not to mention

To accomplish this, an "Action" parameter (Action%) has been added to the calling sequence. Depending on the setting of this variable, the menus can be instructed to operate in a number of different ways.

The Action variable has six different possible settings which tell PullDown-G and VertMenuG how they are to behave. Each of the possible Action values is described in detail below.

COMMON state-

accomplished.

a in every module

ngram, Also,

EO,BAS, shown sing COMMON o sets of COM-

n below are the

hould you decide

Action is set to zero, the menu will operate the way you would expect a "normal" menu to work. The underlying screen is first saved, then the menu is displayed, and finally an INKEY\$ loop repeatedly waits for the user to press a key or a mouse button. Once a key or a mouse button has been pressed, the original screen is restored and control is returned to the The Choice variable(s) may then be examined to see what selection the user chose. a "normal" menu to work. calling program.

When Action is set to 1, both PullDownG and VertMenuG save the screen and display themselves. Control is then returned to the calling program immediately and Action will be set to 3 for subsequent calls. Since Action 3 is how you will be polling the menu subsequently, this saves you an extra Setting Action to 2 lets you re-display the menu in those cases where it may have been overwritten by another possibly overlapping menu. Action 2 also resets itself to 3 for subsequent calls.

> sshown above. what we have

polled to see if a key or button has been pressed. If Action is still set to If the menus are called with Action 3, the keyboard and mouse are merely 3 when the menu returns, it means that no keys or mouse buttons were

Introduction

pressed Escape. In this case, the Choice, Menu, and Ky\$ variables should be examined. Look at the programs DEMOPULL.BAS and DEMOVERT.BAS to see how these variables can be examined. pressed. If Action is returned set to 4, the user either made a selection or

itself and restore the original screen. The last Action value is 5 which tells VertMenuG or PullDownG to remove

means that it is up to you to save and restore each screen in succession manually, except for the last one. that was saved first will be destroyed when the menu is called again. This If you intend to create stacked menus, you should be aware of one important point. Because each menu saves its own underlying screen, the screen

G and VertMenuG in a typical programming context. DEMOMENU. BAS provides complete demonstration of using PullDown-

Sep

Demo Programs

scren, the screen creen in succession

of using PullDown-

SOFTWARE, INC.

A Sugar

Demonstration Programs

Chapter 2:

Demo Programs CHEKKENT SUFTWARE INC. DEMOCURS.BAS or not the cursor is DEMONSTR DEMOCURS.BAS positioning within a p DEMOCIRC.BAS ABOUTPCXBAS DEMOBYTE.BAS DEMOBOX.BAS when you want to u Also, the CircleVE your programs will l when creating progr Workshop. Use th DEMOCIRC.BAS & boxing routine for PositionBox, is de the same on routine can b. a fa DEMOBYTE.BAS DEMOBOX.BAS pixels, draw patter.
The DrawByt VE these corners, use for one of two co using the cursor ke ABOUTPCX BAS ABOUTPCX BAS header. It is an e header monitor. particular monitor. particular and suggests size, and suggests

DEMONSTRATION PROGRAMS

ABOUTPCX.BAS

ABOUTPCX. BAS displays all the information contained in a .PCX file's It is an excellent tool for testing the use of a .PCX file on a ABOUTPCX tells the version of the file, the image size, and suggests which type of monitor it was created on. particular monitor.

Demo Programs

DEMOBOX.BAS

DEMOBOX.BAS shows the operation of the exclusive-oring (XOR) boxing routine for defining a region on the screen. A complete routine, PositionBox, is demonstrated here. This routine accepts keyboard input using the cursor keys to adjust the size of the box. Adjustments are made for one of two corners, upper-left and lower-right. To toggle between these corners, use the SpaceBar.

DEMOBYTE.BAS

The DrawByteVE routine can draw up to 8 pixels at a time, but all using the same color. If these requirements fit your needs, then the DrawByteVE DEMOBYTE.BAS shows how to use the DrawByteVE routine to display pixels, draw patterns, simulate the LINE statement in BASIC and more. routine can be a fast way to draw images.

DEMOCIRC.BAS

DEMOCIRC.BAS shows the use of the CircleVE routine within Graphics Workshop. Use this routine to emulate the BASIC CIRCLE statement when creating programs that don't use the BASIC SCREEN statement, or Also, the CircleVE routine does not require floating point arithmetic so when you want to use its OR, AND, or XOR logical operation abilities. your programs will be smaller.

DEMOCURS.BAS

in CURSOR.BAS. The routine DrawCursor XORs a graphics cursor at DEMOCURS.BAS shows the use of the graphics cursor routines contained the position you specify. A value is maintained that keeps track of whether or not the cursor is currently visible. DrawCursor also demonstrates positioning within a proportional font string.

DEMODIGI.BAS

resolution. is used on. These routines take a portion of the screen and lower the apparent DEMODIGI. BAS shows the screen digitizing routines in DIGITIZE BAS. This can be used to create an artistic effect for the screen it

DEMOEDIT.BAS

routine to accomplish printing the text information within the string. routine to implement a graphics mode cursor, and uses the GPrintOVE field editor within graphics mode. DEMOEDIT.BAS shows how the GEditor routine can provide a complete The routine uses the DrawCursor

DEMOINTR. BAS

DEMOINTR. BAS

QuickSegue demo

DEMOHERC. BAS

DEMOHERC.BAS

Hercules monitor '

DEMOGWBAS

mirror image of a

DEMOGW BAS de

and gives a variety

DEMOEURO.BAS

are used in European languages ASCII character set. Many of the characters in the extended character set Workshop fonts systems with characters extending through the entire DEMOEURO.BAS shows how to use the GraphPak and Graphics

DEMOFADE.BAS

DEMOLTS.BAS

static image. background effect a movie clicker to

pixels are arranged in the form of a 2x2 square. Two uses of the FadeEGA routine are demonstrated brings an image to the visible screen randomly, 4 pixels at a time. The 4 routine is the most complex of the graphics transfer routines. FadeEGA DEMOFADE.BAS shows the FadeEGA routine in action. The FadeEGA

DEMOFONT.BAS

is the use of multiple fonts on the same screen. GraphPak may be drawn at any size, and at any angle. Also demonstrated Both the Vector Fonts described in Chapter 6 and the fonts provided from DEMOFONT BAS presents all the fonts available in Graphics Workshop.

DEMOMOUS.BAS

feature of the PullDo MsgBoxG routines DEMOMENU.BAS DEMOMENU.BAS

up the display to

DEMOLTS. B. MenuG and

DEMOFX.BAS

routine is used by a BASIC subroutine to split the image and move to the by BASIC subroutines to slide images onto the screen. The GMove3VE for moving images on and off the screen. The GMove1VE routine is used DEMOFX.BAS demonstrates the GMove1VE and GMove3VE routines

CHENERY SUFFRANCE INC

Graphics Workshop.

emonstrates the abil

when programing It demonstrates the pr DEMONOUS BAS S

fer for the streng

The GMove1VE routine can also be used to create a mirror image of an image on the screen. screen in two parts.

DEMOGW.BAS

DEMOGW. BAS demonstrates many of the routines in Graphics Workshop, and gives a variety of methods for using them.

DEMOHERC.BAS

n provide a complete es the DrawCursur

thin the string.

DEMOHERC.BAS shows the graphics primitives that can be used on a Hercules monitor without requiring the use of the TSR MSHERC.COM.

DEMOINTR.BAS

oak and Graphics inded character set

QuickSegue demo (QSEGUE.BAS). Two interludes are shown. One uses a movie clicker to display a title. The other provides an interesting DEMOINTR.BAS shows the "interludes" available for adding to the a movie clicker to display a title. The other provides an interesting background effect which can be placed behind titles to add motion to a static image.

DEMOLTS.BAS

n. The FadeEGA at a time. The 4

tines. FadeEGA

DEMOLTS.BAS demonstrates the two Lotus-style menu programs, Lts-MenuG and Lts2MenuG, which come with graphics workshop. DEMOLTS.BAS demonstrates how to initialize the menu arrays and set up the display to use the routine.

DEMOMENU. BAS

DEMOMENU.BAS shows the use of the PullDownG, VertMenuG, and The multi-tasking feature of the PullDownG and MsgBoxG routines are implemented here. MsgBoxG routines together as a complete system.

> shies Workshop. s provided from

so demonstrated

re3VE routines he GMove3VE nd move to the routine is used

DEMOMOUS.BAS

DEMOMOUS.BAS shows the use of mouse routines for graphics mode. It demonstrates the proper use of the HideCursor and ShowCursor routines demonstrates the abilities of all of the other mouse routines provided with when programming for mouse support in the graphics modes. Graphics Workshop.

DEWOSCRL BAS

DEMOMOVE.BAS

and PUT statements in BASIC; all work is done entirely in video memory. GMove2VE routines do not require setting up a large array like the GET a major speed improvement, DEMOMOVE.BAS shows how to use the GMove1VE and GMove2VE images or manipulate images in almost any fashion. The GMove1VE and routines to duplicate images, simulate the GET and PUT in BASIC with and how subroutines can be created to flip

DEMOSCRN.BAS

each other to form of the demo, four are scrolled simulta DEMOSCRL. BAS

DEMOPAL.BAS

of a presentation. It also demonstrates how to simulate motion using the palette. Finally, it demonstrates the need for a proper set of colors when DEMOPAL BAS shows how using the palette can increase the impression creating a real-life image.

DEMOSHAD.BAS

swapping between all of the Graphics DEMOSCRNIBAS

DEMOSHAD.BAS

that provide a flowir

DEMOPLMS.BAS

which is used to save and restore parts of the graphics screen which lie background color, which is not possible with BASIC's PRINT statement beneath the pull-down menus while running in graphics mode. PullDnMSG menu system for graphics mode. It allows the text to have a down menu system. DEMOPLMS.BAS shows the use of a Windows(tm)-like graphics pull-The routine GPrintOVE is a vital part of the Another vital routine is GMove2VE

DEMOVERTBAS

to add a backgroun

DEMOPULL.BAS

not possible with BASIC's PRINT statement while running in restore parts of the graphics screen which lie beneath the pull-down menus. mode. Another vital routine is GMove2VE, which is used to save and graphics mode. It allows the text to have a background color, which is routine GPrint0VE is a vital part of the PullDownG menu system for DEMOPULL.BAS shows the use of graphics pull-down menus. graphics

FONT64.BAS

restore the graph

not possible

mode. Anoth graphics mode The routine GPrin

DEMOVERT.BAS

DEMOSAVE.BAS

assembly and is therefore very fast. screen into the .PCX file format. The routine SavePCXVE is written in DEMOSAVE.BAS shows the use of SavePCXVE for storing a graphics

CWDEMO.BAS

fonts as a base font w and provides an exce fonts. It demonstrat FONT64.BAS is a 1

GNDENO.BAS is th

CHECCHI TOPPINGE INC

paring the speed impr It includes an examp

introduces some diffe

The state of the s

INTERNACIONAL COMPANIONE COMPANIO

Company of the second

De Office VE and e arm like the GET el in de nemon.

DEMOSCRL.BAS

DEMOSCRL.BAS shows the GScrollVE routine in action. Four windows are scrolled simultaneously with and without a delay. In the next portion of the demo, four items placed on the screen are moved inward toward each other to form one object in the middle.

Demo Programs

DEMOSCRN.BAS

rease the impression ate motion using the a set of colors when

DEMOSCRN.BAS shows how the KeepData% parameter available with all of the Graphics Workshop Screen mode routines can be used when swapping between graphics mode to text mode and back again.

DEMOSHAD.BAS

that provide a flowing transition of color to any window. Use these routines to add a background to demonstration images or titles. DEMOSHAD.BAS shows the Shade Vertical and Shade Horizontal routines

DEMOVERT.BAS

wital part of the

vs the text to have a -like graphics pull.

s PRINT statement ine is GMove2VE. S Screen which lie

DEMOVERT.BAS shows the use of graphics vertical scrolling menus. The routine GPrintOVE is a vital part of the VertMenuG menu system for It allows the text to have a background color, which is not possible with BASIC's PRINT statement while running in graphics Another vital routine is GMove2VE, which is used to save and restore the graphics screen which lies beneath the vertical scrolling menus. graphics mode.

FONT64.BAS

menu system for

id color, which is

wn menus. The

used to save and

Jull-down menus

naing in graphics

FONT64.BAS is a font editor written in QuickBASIC to create vector fonts. It demonstrates use of the pull-down menu system for graphics, and provides an excellent way to create new fonts. It accepts GraphPak fonts as a base font which you can outline to form new fonts.

GWDEMO.BAS

GWDEMO.BAS is the demo program the Graphics Workshop demo disk. It includes an example of "mousable" three-dimensional buttons, and introduces some different fade types. Benchmarks are also shown comparing the speed improvements over BASIC's equivalent routines.

(VE is written in

toring a graphics

T SOFTWARE DAY

QSEGUE.BAS

QuickSegue script language is described in detail in Chapter 5. file, used to load graphics files and send them to the screen. This program called QuickSegue is a complete graphics slide show program. It accepts a script consisting of commands from an ASCII input

SCRNDUMP.BAS

to perform rudimentary scaling. SCRNDUMP.BAS shows the print routine ScrnDump2 in action. This routine can print any graphics screen to either an EPSON or compatible dot-matrix, or to a Hewlett Packard LaserJet in either Portrait or Landscape mode. If you're printing to the LaserJet, there is also an option

VIEWPCX.BAS

VIEWPCX.BAS loads a .PCX graphics file, determines the screen mode used to store the screen (EGA, VGA, CGA, or HERC), and then sends the graphics file to the screen. Pressing any key returns to DOS. The the palette accordingly. program also reads the palette information from the .PCX file and adjusts

Assembly Routines Chapter 3: Assembly Routines Control of the second of the s The in action. This SON or compatible letter Portrait or the is also an option tes the screen mode

C), and then sends

urns to Dos. The

CX file and adjusts T SOFTWARE, DIC.

Assembly Routines CRECCENT SOFTWARE INC. SECTION 1: PC) A PCX file is a gr A PCX file is a gr PaintBrash. The I PaintBrash. Into PCX images. Appendix A. Info in Appendix B. virtually all virtually and of virtually and Virtually all of the assembly routines rely on some support material. The support material is generally shared data locations in memory. Data like otherwise would be duplicated for each routine are contained in the file GWVARS.ASM. The assembler file GWVARS.ASM holds the common in GWLLB and GWY.LIB and dWT.LIB and dWT.LIB and dWT.LIB and will be required by most programs created with Graphics Workshop.

SECTION 1: PCX FILES AND PALETTES

A PCX file is a graphics file format created by ZSoft, the makers of PC The routines in this section are used to access and display . Information about the PCX file format is contained in PCX images. Information about the PCX tile format is contained in Appendix A. Information about the Palette and how it is used is contained in Appendix B. PaintBrush.

Assembly Routines

Assembly Routines

DispPCXC

Assembler subroutine contained in GW.LIB

Purpose:

displays the image to a CGA specified video page DispPCXC continues the loading process started by OpenPCXFile% and

Syntax

CALL Dispical I

DispPCXH continued displays the image

Where:

■ Syntax:

CALL DispPCXC (BYVAL VideoPage%)

Where:

Assembly Routines

VideoPage% is 0 for the default first display page (Visual Display Page). A value of 1 specifies the second display page.

Comments:

speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation All parameters for this routine are passed by value to provide the maximum

speed. Including the or modules which I

VideoPage® is 0

A value of 1 specific Comments:

All parameters for t

The function Open file and loads the he

PCX file is interne

PCX file is intended for. file and loads the header information to determine which screen mode the The function OpenPCXFile% must be called first, as it opens the PCX

2-color mode, will translate the image into a tiled 2-color image. Due to the nature of the CGA's video memory, this routine will work on both the CGA 2-color graphics mode and the CGA 4-color graphics mode. Loading a CGA 4-color graphic image while the computer is in the CGA

program and the BASIC routine DisplayPCXFile. Use of this routine is demonstrated in the VIEWPCX.BAS example

See Also:

OpenPCXFile%

OpenPCXFile%

affords full company
Use of this routine
Program and the BA

does not require in

The Hercules (or with the use 3 statement _____ Graphics Worksh

CHECKENT SUFFIMALE, INC.

DispPCXH

subroutine contained in GW.LIB Assembler

Purpose:

DispPCXH continues the loading process started by OpenPCXFile% and displays the image to a Hercules specified video page.

Syntax:

CALL DispPCXH (BYVAL VideoPage%)

Where:

sual Display Page).

VideoPuge% is 0 for the default first display page (Visual Display Page). A value of 1 specifies the second display page.

Assembly Routines

Comments:

vide the maximum ng of all programs

Deration.

it opens the PCX 1 Screen mode the ine will work on er is in the CGA

X. BAS example

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The function OpenPCXFile% must be called first, as it opens the PCX file and loads the header information to determine which screen mode the PCX file is intended for.

The Hercules screen mode is of an odd size, 720x348 pixels when created with the use of MSHERC COM. You can replace the BASIC SCREEN 3 statement in your program with the Screen3 routine provided with the Graphics Workshop to set the Hercules into graphics mode. This method does not require the use of the TSR program MSHERC, COM, yet it affords full compatibility. Use of this routine is demonstrated in the VIEWPCX.BAS example program and the BASIC routine DisplayPCXFile.

See Also:

OpenPCXFile%

SOFTWARE, DIC

DispPCXVE

Assembler subroutine contained in GW.LIB

Purpose:

DispPCXVE continues the loading process started by OpenPCXFile% and displays the image to a VGA or EGA specified video page.

Syntax

CALL DISPORAVA

DispPCXV256 co

Syntax:

CALL DispPCXVE (BYVAL VideoPage%)

Where:

Assembly Routines

A value of 1 specifies the second display page. VideoPage% is 0 for the default first display page (Visual Display Page).

Comments:

or modules which use this routine will insure proper operation speed. Including the file GWDECL.BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

or modules which

All parameters for speed. Including t

A value of I speci

VideoPage% is 0

Comments:

Where:

PCX file is intended for. file and loads the header information to determine which screen mode the The function OpenPCXFile% must be called first, as it opens the PCX

and VGA 2-color graphics modes do not utilize the plane scheme most of the other EGA and VGA screens use. The DispPCXVE routine will still out the intensity plane). This routine will load those files properly as well. version 2 saves only three of the four graphics planes in the file (leaving work equally as well on those. Also, note that PaintBrush for Windows utilize the plane system created for EGA and VGA graphics. The EGA modes is identical. This routine works for all the screen modes which SCREEN 7, 8, 9, 11, and 12 since the video memory for all of these The DispPCXVE routine works equally well when BASIC is operating in

See Also:

program and the

The function Open file and loads the h PCX file is intende

program and the BASIC routine DisplayPCXFile Use of this routine is demonstrated in the VIEWPCX.BAS example

See Also:

OpenPCXFile%

CRESCRIT SUFFMARE, INC.

DispPCXV256

Assembler subroutine contained in GW.LIB

Purpose:

Obsulto Ar left and

DispPCXV256 continues the loading process started by OpenPCXFile% and displays the image to a VGA 256-color mode specified video page.

Syntax:

CALL DispPCXV256 (BYVAL VideoPage%)

sual Display Pagel.

VideoPage% is 0 for the default first display page (Visual Display Page). A value of 1 specifies the second display page.

Assembly Routines

Comments:

Nidethemaximum ng of all programs

peration,

it opens the PCX h screen mode the IC is operating in y for all of these

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

file and loads the header information to determine which screen mode the The function OpenPCXFile% must be called first, as it opens the PCX PCX file is intended for. Use of this routine is demonstrated in the VIEWPCX.BAS example program and the BASIC routine DisplayPCXFile.

OpenPCXFile%

X.BAS example

ush for Windows properly as well.

the file (leaving

scheme most of

GetPaletteVGA%

function contained in GW.LIB Assembler

Purpose:

up the specified pa GetPalTriple VGA

Purpose

are using a VGA video adapter. This routine is used for the EGA and VGA screen modes, but only if you GetPaletteVGA % returns the current value of the palette register specified

Syntax:

Colr% = GetPaletteVGA%(BYVAL PalRegister%)

Where:

Syntax

CALL GetPall

Where:

Assembly Routines

(use 0 through 15). PalRegister% is one of the 16 available colors for the EGA screen modes

color value from the palette. Colr% is the value returned and is between 0 and 63 which represents the

Comments:

any problems as it contains a declaration for this function. the file GWDECL.BAS at the beginning of all your programs will avoid parameter by value, it must be declared before it can be used. Including Because GetPaletteVGA% has been designed as a function and passes a

> speed. Including th One parameter for

or modules which

To get the pale

the color planes to

Comments:

Red%, Green%,

resolution screens PalRegister% spe

modes (use 0 throu

The short example below shows how to get the palette value of color 1 on VGA monitor.

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's 'sets monitor in a VGA mode

PaletteValue% = GetPaletteVGA%(1)

a method of re-mapping the EGA's palette the current palette settings. One, if you are setting the palette registers, The EGA video registers are write-only. There is no way to read the EGA article in the March 1990 issue of Programmer's Journal which discussed what is the value of each palette register. Two, there was an excellent you can maintain a list of what values you have set. EGA palette value is not a simple task. There are two options to determine registers which contain the palette information. Therefore, getting an Then you will know

SetPalTripleVGA, Ha See Also: two bits to the right.

routine HandlePCXP Palette values read fi

GetPallripleNSA(3

-3001DILL\$,

See Also:

SetPaletteEGA

CREACENT SOFTWARE, INC.

Thes is OWIR

GetPalTripleVGA

subroutine contained in GW.LIB Assembler

Purpose:

odes, but only if my Te terme number

GetPalTripleVGA returns the Red, Green, and Blue values which make up the specified palette. This routine can be used only in the VGA screen

Syntax:

CALL GetPalTripleVGA(BYVAL PalRegister%, Red%, Green%, Blue%)

EGA screen modes

which represents the

PalRegister% specifies one of the 256 available colors for the VGA screen modes (use 0 through 255), or one of the 16 available colors for the higher resolution screens (use 0 through 15).

Assembly Routines

Red%, Green%, and Blue% return values between 0 and 63 for each of the color planes for that palette register.

Comments:

ction and passes a e used. Including Ograms will awid value of color 1 on

OII.

One parameter for this routine is passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

To get the palette information for the color 3 on a VGA screen mode, use the following code fragment:

'\$INCLUDE: 'GWDECL.BAS' SCREEN 13 DEFINT A-Z

'sets monitor in a VGA mode 'required for BYVAL's

GetPalTripleVGA(3, Red%, Green%, Blue%)

Palette values read from .PCX files are shifted two bits to the left. The routine HandlePCXPalette does an integer divide by 4 to shift the values two bits to the right.

See Also:

efore, getting an y to read the EGA

ions to determine en you will know

palette registers,

SetPalTripleVGA, HandlePCXPalette

was an excellent which discussed T SOFTWARE, INC

GWFileSize&

function contained in GW.LIB Assembler

OpenPCXFile & C

information, Including

Arrays = SPACES

Purpose:

GWFileSize& will quickly return the length of a named file.

Syntax:

Size& = GWFileSize& (Filename\$)

Where:

Filename\$ is the name of the file.

100

Where:

Arrays is a string Filenames is a str

header information

Assembly Routines

Size& receives its length in bytes. If the file does not exist Size& is instead assigned a value of -1.

Comments:

of all your programs will avoid any problems as it contains a declaration Because GWFileSize& has been designed as a function, it must be declared before it can be used. Including the file GWDECL.BAS at the beginning for this function.

DEMOSHAD.PCX. The brief example that follows shows how to find the file size of the file

DispPCXVE, Disp

The following

After opening the P

can be used to

Comments:

for the 256-color n tion contained in th

the PCX file

Arrays = 37.25

Size& = GWFileSize& ("DEMOSHAD.PCX")

file size of .PCX files. The section on the routine QBinaryLoad% goes into more detail of the use of this routine. This product's main purpose for using GWFileSize& is to determine the

QuickPak Professional. GWFileSize& does not use the critical error handler that comes with FileSize& routine. Those users of QuickPak Professional will note that QuickPak has a The difference between the two routines is that

See Also:

QBinaryLoad%

CHECKETT SUFFWARE INC.

program and the BAS Use of this routine

ELSEIF WhichScreen ELSEIF WhichGorean CALL Handle FLXPale

CHL WhichPCKSorp

OpenPCXFile%

CAMB

Assembler function contained in GW.LIB

Purpose:

OpenPCXFile % opens the specified PCX file, and loads the header information, including palette information, into the string specified.

Syntax:

Array\$ = SPACE\$(68 + 768)
Success% = OpenPCXFile%(Filename\$, Array\$)

exist Sizely is instead

Filename\$ is a string containing the filename of the PCX file.

Assembly Routines

Array\$ is a string of length (68 + 768). The first 68 bytes receive the header information. If the file is a 256-color PCX file, then the information contained in the last 768 bytes of this string are the palette information for the 256-color mode.

Comments:

AS at the beginning ntains a declaration

i must be declared

file size of the file

is to determine the maryLoad% goes

After opening the PCX file with this function, one of the following routines can be used to display the PCX file: DispPCXC, DispPCXH, DispPCXVE, DispPCXV256. The following code fragment will load and display a PCX file whose name is contained in Filenames. It assumes that the correct screen mode for the PCX file has already been set.

Array\$ = SPACE\$(68 + 768)
IF NOT OpenPCXFile%(Filename\$, Array\$) THEN EXIT SUB

OuickPak has a routines is that that comes with

CALL WhichPCXScreen(Array\$, WhichScreen%)

CALL HandlePCXPalette(Array\$, WhichScreen%)

IF WhichScreen = 4 OR WhichScreen = 6 THEN

CALL DispPCXC(VideoPage%)
ELSEIF WhichScreen = 2 THEN
CALL DispPCXH(VideoPage%)

ELSEIF WhichScreen = 9 THEN

CALL DispPCXV256(VideoPage%)

CALL DispPCXVE(VideoPage%)

CALL ClearVE

Use of this routine is demonstrated in the VIEWPCX.BAS example program and the BASIC routine DisplayPCXFile. END IF

CRESCENT SOFTWARE, INC.

NT SOFTWIRE, DK.

PCXATTAYC

Purpose:

screen specified by PCXATTYC takes

Where:

CHEL ACKARTAGE (

See Also:

Assembly Routines

DispPCXC, DispPCXH, DispPCXVE, DispPCXV256, DisplayPCXFile, HandlePCXPalette, WhichPCXScreen

■ 3 - 10

CRESCENT SOFTWARE, INC.

SECRET SOFTWARE, INC.

See Also;

An example of using the OSEGUE.BAS.

2-color mode will tra

Due to the nature of both the CGA 2-color Loading a CGA 4-col

the QBinaryLr

using the PCXArray

a binary file into ¿ The Array % () must ! Both parameters fo

Comments: A value of 1 specifi VideoPage% is 0 fo of the array and ext complete .PCX file Array Seg % is the

operation. all programs or m maximum speed. It

QBinaryLoad%

PCXArrayC

Assembler subroutine contained in GW.LIB

Purpose:

PCXArrayC takes an array containing a PCX file and sends it to the CGA screen specified by video page.

■ Syntax:

CALL PCXArrayC (BYVAL ArraySeg%, BYVAL VideoPage%)

■ Where:

ArraySeg% is the segment of an integer array which already holds the complete .PCX file, starting at the memory location of the first element of the array and extending to the last memory location of the array.

Assembly Routines

VideoPage% is 0 for the default first display page (Visual Display Page).

A value of 1 specifies the second display page.

Comments:

Both parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The Array %() must be loaded with the QBinary Load % routine which loads a binary file into a specific memory location. A complete example of using the PCXArrayX routines is shown in context in the comments for the QBinaryLoad% routine.

Due to the nature of the CGA's video memory, this routine will work on both the CGA 2-color graphics mode and the CGA 4-color graphics mode. Loading a CGA 4-color graphic image while the computer is in the CGA 2-color mode will translate the image into a tiled 2-color image.

An example of using this routine is contained in the demonstration program OSEGUE.BAS.

See Also:

QBinaryLoad%

ENT SOFTWARE, INC.

The Graphics Workshop

PCVATTATVE take

Purpose:

Hercules screen specified by video page. PCXArrayH takes an array containing a PCX file and sends it to the

Syntax:

CALL PCXArrayH (BYVAL ArraySeg%, BYVAL VideoPage%)

Where:

CALL PENTATE

ArraySeg % is the

of the array and ex complete .PCX file

Where:

of the array and extending to the last memory location of the array. complete .PCX file, starting at the memory location of the first element ArraySeg% is the segment of an integer array which already holds the

A value of 1 specifies the second display page. VideoPage% is 0 for the default first display page (Visual Display Page).

all programs or modules which use this routine will insure proper Both parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of

all programs or m

maximum speed.

Both parameters for

Comments:

A value of I specif VideoPage% is 0 t

the QBinaryLoad% routine. using the PCXArrayX routines is shown in context in the comments for a binary file into a specific memory location. A complete example of The Array % () must be loaded with the QBinary Load % routine which loads

Screen3 routine provided with Graphics Workshop to set the Hercules into can replace the BASIC SCREEN 3 statement in your program with Hercules mode. with the use of MSHERC.COM. Rather than using MSHERC.COM, you The Hercules screen mode is of an odd size, 720x348 pixels when created MSHERC.COM, but would allow a program to display a PCX file in the graphics mode. This method does not require the use of the TSR program See the routine Screen3 for details and restrictions.

The PCXArra

the QBinaryLo~ 1%

a binary file into 2 The Array%() must

using the PCXArr,

modes is identical. in SCREEN 7, 8, o

QSEGUE.BAS An example of using this routine is contained in the demonstration program

QBinaryLoad%, HercThere%, Screen3

CREECENT SOFTWARE, INC.

See Also: QBinaryLoad%

QSEGUE.BAS. An example of using t out the intensity plane

version 2 saves only t work equally as well the other EGA and Vo and VGA 2-color gra utilize the plane syste

PCXArrayVE

subroutine contained in GW.LIB Assembler

Purpose:

PCXArrayVE takes an array containing a PCX file and sends it to the EGA or VGA screen specified by video page.

Syntax:

CALL PCXArrayVE (BYVAL ArraySeg%, BYVAL VideoPage%)

Where:

on already holds the of the first element

n of the array.

Stral Display Page.

ArraySeg% is the segment of an integer array which already holds the complete. PCX file, starting at the memory location of the first element of the array and extending to the last memory location of the array.

VideoPage% is 0 for the default first display page (Visual Display Page). A value of 1 specifies the second display page.

Comments:

lue to provide the at the beginning of

Both parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL BAS at the beginning of all programs or modules which use this routine will insure proper

The Array %() must be loaded with the QBinaryLoad routine which loads a binary file into a specific memory location. A complete example of using the PCXArrayX routines is shown in context in the comments for the OBinaryLoad% routine.

in SCREEN 7, 8, 9, 11, and 12 since the video memory for all of these modes is identical. This routine works for all the screen modes which The PCXArrayVE routine works equally well when BASIC is operating and VGA 2-color graphics modes do not utilize the plane scheme most of utilize the plane system created for EGA and VGA graphics. The EGA the other EGA and VGA screens use. The PCXArrayVE routine will still Also, note that PaintBrush for Windows version 2 saves only three of the four graphics planes in the file (leaving out the intensity plane). This routine will load those files properly as well. work equally as well on those.

An example of using this routine is contained in the demonstration program OSEGUE.BAS.

See Also:

OBinaryLoad%

CRESCENT SOFTWARE, INC.

NT SOFTWARE, DK.

coutine which loads inplete example of

will insure proper

the comments for

the TSR program a PCX file in the program with the the Hercules into

nstration program

d restrictions.

3 - 13

PCXArrayV256

Assembly Routines

subroutine contained in GW.LIB Assembler

Purpose:

PCXArrayV256 takes an array containing a PCX file and sends it to the VGA screen running in 256-color mode.

Syntax:

CALL PCXArrayV256 (BYVAL ArraySeg%, BYVAL VideoPage%)

Where:

ColStart% is a colu LineStart % is a val

Comments:

Symtax

CALL PositionPCXI

an EGA or VGA his

OpenPCXFile.

Position PCX TVE set

complete .PCX file, starting at the memory location of the first element of the array and extending to the last memory location of the array. ArraySeg% is the segment of an integer array which already holds the

mode. VideoPage% is 0 for the default first display page (Visual Display Page). Memory limitations allow only one page on the VGA when in 256-color

Comments:

all programs or modules Both parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of which use this routine will insure proper

DisplayPCXFile2. Proper usage of the operation. all programs or m maximum speed. I Both parameters to

the QBinaryLoad% routine. using the PCXArrayX routines is shown in context in the comments for a binary file into a specific memory location. A complete example of The Array % () must be loaded with the QBinaryLoad % routine which loads

QSEGUE.BAS. An example of using this routine is contained in the demonstration program

See Also:

QBinaryLoad%

CRESCENT SOFTWARE, INC.

PositionPCXVE

subroutine contained in GW.LIB Assembler

Purpose:

PositionPCXVE sets the screen location to place the next PCX image on an EGA or VGA high-resolution screen mode. This routine is called after OpenPCXFile.

Syntax:

arear holy the ार्टी कीर विज्ञा हो होता है।

त of the बन्दा,

(Sual Display Page). When in 256-color

CALL PositionPCXVE (BYVAL LineStart%, BYVAL ColStart%)

Where:

LineStart% is a value between 0 and 479 for a VGA display.

Assembly Routines

ColStart% is a column number between 1 and 80.

Comments:

Both parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

> lue to provide the at the beginning of will insure proper

routine which loads mplete example of

ristration program

Proper usage of this routine is shown in the example subroutine DisplayPCXFile2.

CRESCENT SOFTWARE, INC.

VT SOFTWARE, INC.

3 - 15

QBinaryLoad%

Assembler function contained in GW.LIB

See Also:

that the surroun mon

Purpose:

If the file is read successfully, this function returns a non-zero value. the integer array so that it can be displayed on the screen at a later time. QBinaryLoad% loads the PCX graphics file specified by Filename\$ into

Syntax:

Success = QBinaryLoad%(Filename\$, SEG Array%)
IF NOT Success THEN PRINT "File Error"

Where:

Assembly Routines

be used here. Filename\$ is a string containing any valid file name. No wildcards should

two-dimensional array is required if you intend to load files larger than in the file. The array can be one-dimensional or two-dimensional. Array%0 is an integer array with half as many elements as there are bytes

Comments:

screen modes: A method for loading a PCX file to the EGA or VGA high-resolution

```
DEFINT A-Z

SINCLUDE: 'GWDECL.BAS'

SCREEN 9

Filename$ = "Graph1.PCX"

Size& = GWFile5ize&[Filename$]

REDIM Array%(Size& \ 4 + 1, 1)

Success = QBinaryLoad(Filename$, SEG Array%(0, 0))

IF Success THEN

ArraySeg% = VARSEG(Array%(0, 0))

CALL PCXArrayVE(ArraySeg%, 0)

ELSE

PRINT "Error Loading File"
```

PCXArrayH, or PCXArrayV256 routines, simply replace the above line containing the call to PCXArrayVE. EGA or VGA high-resolution screen modes. To use the PCXArrayC, The above code has been simplified to expect the PCX file to be for the

END IF

WhichPCXScreen will not work directly, because it is designed for the the screen mode a PCX file was created on. Note that the code in Code like that in the routine WhichPCXScreen can be used to determine

RESCENT SOFTWARE INC.

Assembly Routines

PCX display routines like DispPCXVE which do not load the entire file into memory before displaying it. Furthermore, when using the above code fragment for the array method of displaying PCX files, it is likely that the screen mode will be known ahead of time.

See Also:

No. of Street, or other Persons and Street, o The state of the s

C. Day-Car. Valle.

GWFileSize&, PCXArrayVE, PCXArrayC, PCXArrayH, PCXArray-

Assembly Routines

us as there are bytes ad files larger than

0-dimensional

A high-resolution

CRESCENT SOFTWARE, INC.

file to be for the e the above line he PCXArrayC.

ed to determine nat the code in esigned for the SOFTWARE, INC

3 - 17

SavePCXVE

subroutine contained in GW.LIB **Assembler**

Purpose:

SavePCXVE saves the current EGA or VGA high-resolution screen to the .PCX file specified by Filename\$.

Syntax:

CALL SavePCXVE(Filename\$)

Assembly Routines

used, then the PCX file will not load properly using the DisplayPCXFile routine supplied with Graphics Workshop. Filename\$ must contain the extension ".PCX". If the extension is not Filename\$ is a string which holds the name of the .PCX file to be saved.

Where:

Syntax larger palette from SetPal Triple VGA re SetPaletteEGA sets Purpose:

CALL SetPaletteEb

routine should be

(use 0 through 15). PalRegister% is on

the available colors

Value% is a number

Comments:

Comments:

itself to save the appropriate number of video lines. This routine determines the current EGA or VGA video mode and adjusts

a red circle to the file REDCIRC.PCX The short example below demonstrates saving an EGA screen containing

```
DEFINT A-Z
'$INCLUDE: 'GWDECL.BAS'
CALL SavePCXVE ("REDCIRC.PCX")
                                   CIRCLE (100, 100), 80, 4
                                                  required for BYVAL's set the monitor in the
                                 in the EGA mode 'draw a red circle
```

The code fragm

or modules wh

speed, including to All parameters for t

A demonstration of using this routine can be found in the example program DEMOSAVE.BAS.

successful, and a non-zero value otherwise SavePCXVE% as a function. It will return a 0 if the file operation was statements in GWDECL.BAS which need to be changed if you want to use The routine SavePCXVE can also be used as a function. Their are declare

DispPCXVE, PCXArrayVE

See Also: GerPalette VGA, SerPal

SetPaletteEGA routine

other Graphics Works If you are using the (statement, then using I EGA screen modes: This routine replaces

CALL SetPaletteECA

\$TINCLUDE:

CHESCHILL SOLLINGS

SetPaletteEGA

S A MOVIE

subroutine contained in GW.LIB Assembler

Purpose:

SetPaletteEGA sets the color value for the palette register specified. This SetPalTripleVGA routine supplies the VGA screen modes with a much routine should be used only in the EGA screen modes. larger palette from which to choose.

Syntax:

CX file to be saved. the extension is not the DisplayPCXFile

CALL SetPaletteEGA(BYVAL PalRegister%, BYVAL Value%)

Where:

PalRegister% is one of the 16 available colors for the EGA screen modes (use 0 through 15).

Assembly Routines

Value% is a number between 0 and 63 which is used to represent one of the available colors in the palette.

to mode and adjusts

A Screen containing

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The code fragment below demonstrates how to change the palette for color number 1 to be represented as white on the screen.

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS'

'required for BYVAL's 'set the monitor in an EGA mode

CALL SetPaletteEGA (1, 63)

Their are declare if you want to use The operation was

example program

63 represents white

This routine replaces the PALETTE statement in BASIC for use with the EGA screen modes. If you are not using QuickBASIC's SCREEN statement, then using BASIC's PALETTE statement would cause an error. If you are using the Graphics Workshop Screen9 routine or one of the other Graphics Workshop routines which set the screen mode, use the SetPaletteEGA routine instead of BASIC's PALETTE statement.

GetPaletteVGA, SetPalTripleVGA, Screen9

VGA screen modes This nution replaces

SetPalTripleVGA

subroutine contained in GW.LIB **Assembler**

Purpose:

SetPalTripleVGA sets the red, green and blue values for a specified color in the palette. This routine can be used only in the VGA screen modes.

See Also: SetPaletteEGA, GetP

SeaPaletteEGA routin

other Graphics Word If you are using the statement, then using

Syntax:

CALL SetPalTriplevGA (BYVAL PalRegister%, BYVAL Red%, BYVAL Green%, BYVAL Blue%)

Where:

Assembly Routines

resolution screen modes (use 0 through 15). (use 0 through 255), and one of the 16 available colors for the higher PalRegister% is one of the 256 available colors for the VGA screen modes

the intensity of that color in the overall color generated for this palette Red%, Green% and Blue% are all values between 0 and 63, which specify

Comments:

speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation All parameters for this routine are passed by value to provide the maximum

The code fragment below shows how to set color number 13 to be represented on the screen as white

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's 'sets the monitor into a VGA mode

CALL SetPalTripleVGA (13, 63, 63, 63)

Setting the Red, Green and Blue values to their maximum values (63) will produce the color of white. provide an equal mix of all colors at their brightest intensities, and hence

■ 3 - 22

THE THEORY WAS THE

Shed noming

Volsaren mala.

This routine replaces the PALETTE statement in BASIC for use with the VGA screen modes. If you are not using QuickBASIC's SCREEN statement, then using BASIC's PALETTE statement would cause an error. If you are using the Graphics Workshop Screen13 routine or one of the other Graphics Workshop routines which set the screen mode, use the SetPaletteEGA routine instead of BASIC's PALETTE statement.

■ See Also:

15. SIM

SetPaletteEGA, GetPalTripleVGA, Screen13

te VGA screen modes colors for the higher ated for this patette

rovide the maximum ting of all programs

Operation.

number 13 to be

um values (63) will ensities, and hence

Assembly Routines

CRESCENT SOFTWARE, INC.

ENT SOFTWARE, DIC

3 - 23

Section 2:

BASIC Graphics Replacements

a Purpose:

Circle VE is a replac doesn't require usin

CircleVE

than BASIC's equivalents. These routines also provide more power than their BASIC equivalents. The routines in this section could be used to replace graphics statements like BASIC's PSET. Most of the routines in this section are much faster

Assembly Routines

these are documented in Section 1 of this chapter. There are also equivalent routines for BASIC's PALETTE statement, but

Where:

Colris is the color

Radius% is the rad XCenter% and YC

serves to tell the rou

performed when dra Add one of the valu Syntax:

CALL CirclevE(BYE Radius#, BYEAL

use logical operation

■ 3 - 24

CRESCENT SOFTWARE, INC.

CHANCELL WILLIAMS INC.

an ellipse. The standard equation

Comments: XAspect% and VAspe CircleVE the followin To create a rubber-ba Any other values will

19 + 1 = 1/10)

CircleVE

Sephin Manager

Assembler subroutine contained in GW.LIB

■ Purpose:

doesn't require using floating point arithmetic, but also has the ability to use logical operations like XOR, OR, and AND when drawing the circle. Circle VE is a replacement for the BASIC CIRCLE statement that not only

CALL CircleVE(BYVAL XCenter%, BYVAL YCenter%, BYVAL Radius%, BYVAL Colr%, BYVAL XAspect%, BYVAL YAspect%)

Syntax:

ETTE statement, but

Where:

XCenter% and YCenter% define the center of the circle on the screen.

Assembly Routines

Radius% is the radius of the circle in pixels.

Colr% is the color that the circle will be drawn in. This variable also Add one of the values in the table below to change the logical operation serves to tell the routine which of the logical operations should be used. performed when drawing the circle.

VALUE 0 2048 4096	LOGICAL OPERATION PSET (Replace) OR AND
1110	Ϋ́

Any other values will produce unusual (and probably undesirable) effects.

To create a rubber-banding effect with the color grey, you would pass CircleVE the following color value:

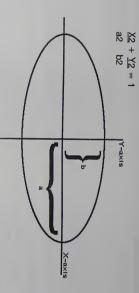
Colr% = 7 + 6144

XAspect% and YAspect% are used to draw ellipses.

Comments:

The standard equation for an ellipse is shown below with an example for an ellipse.

NT SOFTWARE BYC



Syntax:

SHELD THU

The following exam

Comments:

purpose:

The routine CIsVE

If you take the horizontal as your major axis, the distance a will correspond to the Radius% specified above. For the desired b use the following equation to find the value for XAspect%:

CALL LineWE (100, WHILE INCENS = """

STREET 12

CALL CISVE

$$XAspect\% = (32767\% * b) / a$$

in your program. Note that YAspect% would be left as zero. done in the QuickBASIC interpreter, and the result would then be placed This equation could be done using long integer arithmetic at runtime, or

If you flip the ellipse so that the Y-axis is the major axis, then you would simply replace YAspect% into the above equation and leave XAspect% as zero.

See Also:

LineVE

CHACENT SOFTWARE, INC.

CISVE

Assembler subroutine contained in GW.LIB

The routine ClsVE clears the current EGA or VGA video screen. Purpose:

Syntax:

CALL CISVE

Comments:

The following example will clear the VGA video screen after drawing a

ce a will correspond use the following

Assembly Routines

DEFINT A-Z \\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's SCREEN 12 'sets the monitor in V6A mode

CALL LineVE (100, 100, 200, 200, 15) WHILE INKEY\$ = "": WEND

CALL CISVE

retic at runtime, or ould then be placed

S Zero.

is, then you would have XAspect%

CRESCENT SOFTWARE, INC.

T SOFTWARE, INC.

3 - 27

DrawPointV

purpose;

(X, Y) coordinate DrawPointVE draw

DrawPointH

subroutine contained in GW.LIB Assembler

Purpose

require that MSHERC.COM be loaded. coordinate. This routine simulates the BASIC PSET routine, but does not DrawPointH draws a pixel on a Hercules screen at the specified (X, Y)

Syntax:

CALL DrawPointH(BYVAL XPos%, BYVAL YPos%, BYVAL PointColor%)

Where:

CALL DrawPointil

PointColor% is a XPos% and YPos

Comments:

Where:

XPos% and YPos% specify an (X, Y) coordinate on the screen

PointColor% is a color value either 0 or 1.

Comments:

speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation All parameters for this routine are passed by value to provide the maximum

MSHERC.COM TSR program. The following brief example shows how to plot a point at coordinate 100, 100 on a Hercules graphics screen without having to use the

on the VGA so

CALL DrawPointVE

The following Tou

or modules which All parameters for t

speed. Including th

DEFINT A-Z CALL Screen3 '\$INCLUDE: 'GWDECL.BAS' 'set the Hercules into graphics mode 'required for BYVAL's

CALL DrawPointH (100, 100, 1)

show the algorithm for lines and circles. LINEBAS.BAS and CIRCBAS.BAS. Later Workshop will have more support for the Hercules screen mode. The DrawPointH routine can be used by the BASIC example routine which show the algorithm for lines and circles. These routines are Later versions of Graphics

See Also:

DrawByteVE, GetPo See Also: pixel at a time, and DEMOBNCH.BAS DrawPointVE is favo

GetPointH%

ON THE LANGE LANGE

DrawPointVE

aned OWER

the specified (X. Y) notine, busines not

subroutine contained in GW.LIB Assembler

DrawPointVE draws a pixel on an EGA or VGA screen at the specified (X, Y) coordinate. This routine simulates the BASIC PSET routine. Purpose:

CALL DrawPointVE(BYVAL XPos%, BYVAL YPos%, BYVAL PointColor%)

Syntax:

the screen

XPos% and YPos% specify an (X, Y) coordinate on the screen.

PointColor% is a color value from 0 to 15.

Assembly Routines

Comments:

ovide the maximum ing of all programs

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The following code fragment shows how to plot a pixel in the color blue on the VGA screen at location 300, 200.

> mg to use the at coordinate 100

'\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's SCREEN 12 'Sets the monitor in a VGA mode DEFINT A-Z

CALL DrawPointVE (300, 200, 1)

DrawPointVE is faster than QuickBASIC's PSET routine, as shown in the DEMOBNCH. BAS demonstration. DrawByteVE can set more than one pixel at a time, and GetPointVE% will return the color of any pixel.

See Also:

ple routine which se routines are ns of Graphics

DrawByteVE, GetPointVE%

in mode.

time. The GetPo on the screen.

DrawPointVEOpts

Assembler subroutine contained in GW.LIB

See Also: DrawByteVEOpti

■ Purpose:

the screen. use a logical operation like OR, AND, or XOR when placing the pixel on one benefit, the option of using a different logical operation than straight replacement. Like the LineVE and CircleVE routines, this routine can DrawPointVEOpts draws a pixel on an EGA or VGA screen at the specified $(X,\,Y)$ coordinate. This routine simulates the BASIC PSET routine with

Syntax:

Assembly Routines

CALL DrawPointVEOpts (BYVAL XPos%, BYVAL YPos%, BYVAL PointColor%)

Where:

XPos% and YPos% specify an (X, Y) coordinate on the screen.

below to the color, then the associated logical operation will be performed. PointColor% is a color value from 0 to 15. If you add one of the values

VALUE	LOGICAL OPERATION
0	PSET (Replace)
2048	OR
4096	AND
6144	XOR

Any other values will produce unusual (and probably undesirable) effects.

Comments:

speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation. All parameters for this routine are passed by value to provide the maximum

pixel at location 120, 300 of a VGA screen The short example below shows how to use the XOR feature to invert the

DEFINT A-Z

'\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's

SCREEN 12 'sets monitor in a V6A mode

CALL DrawPointVEOpts (120, 300, 15 + 6144)

DrawByteVEOpts routine could be used to set more than one pixel at a routine, as shown in the DEMOBNCH.BAS demonstration. The DrawPointVEOpts routine is faster than the QuickBASIC PSET

CHANGE INC.

The Graphics Workshop

Assembler CANUR

time. The GetPointVE% function will return the color value of any pixel on the screen.

See Also:

DrawByteVEOpts, GetPointVE%

C PUST course with security action than straight to playing the pixel on

d one of the values

the screen.

will be performed.

desirable) effects.

ide the maximum g of all programs ture to invert the

ration

Assembly Routines

CRESCENT SOFTWARE, INC.

KERASIC PSET The Intration.

n one pixel at a

3 - 31

GetPointH%

Assembly Routines

function contained in GW.LIB **Assembler**

Purpose:

GetPointH% returns the color of the pixel at a specified (X, Y) coordinate.

Syntax:

VX = GetPointVEX

dinate.

GetPointVE% retu

Where:

Syntax:

V% = GetPointH%(BYVAL XPos%, BYVAL YPos%)

Where:

XPos% and YPos% make up the (X, Y) coordinate.

V% will receive a color value of either 0 or 1.

Assembly Routines

Comments:

or modules which use this routine will insure proper operation. speed. Including the file GWDECL.BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

All parameters for th

speed. Including th or modules which u

Comments:

EGA and VGA scre V% will receive a XPos% and YPos%

The code fragment below shows how to get the color of the pixel at location 100, 100 on the Hercules graphics screen.

```
DEFINT A-Z
```

CALL Screen3 '\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's

'sets Hercules into graphics mode

V% = GetPointH% (100, 100)

The above code fragment also demonstrates the Screen3 routine which sets the Hercules into graphics mode without the need for the TSR MSHERC.COM.

See Also:

DrawPointH

See Also: DrawPointVE, GetCac

entirely in Appendix E special cache. This There is also a routin

Tole = GetPointVEX

on the EGA bigh-The next code frag

CAN STRIKMAND MATERIAL

GetPointVE%

a ed in GNIIB Assembler

ed /X. Y. condinae.

function contained in GW.LIB Assembler

Purpose:

GetPointVE% returns the color of the pixel at a specified (X, Y) coor-

Syntax:

V% = GetPointVE%(BYVAL XPos%, BYVAL YPos%)

Where:

XPos% and YPos% make up the (X, Y) coordinate.

V% will receive a color value between 0 and 15 for the high-resolution EGA and VGA screen modes.

Assembly Routines

Comments:

Ovide the maximum ling of all programs

Operation.

the pixel at location

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The next code fragment shows how to get the color at a specified location on the EGA high-resolution graphics screen.

'SINCLUDE: 'GWDECL.BAS' 'required for BYVAL's SCREEN 9 'sets the monitor in EGA mode

Colr% = GetPointVE% (200, 300)

need for the TSR an3 routine which

There is also a routine for getting the color of a pixel which utilizes a special cache. This routine is called GetCacheVE% and is contained entirely in Appendix E.

See Also:

DrawPointVE, GetCacheVE%

GPrint0C2

subroutine contained in GW.LIB Assembler

Purpose

the specified color. GPrint0C2 prints a string on the 2-color CGA high-resolution screen in

Syntax:

CALL GPrint0C2 (BYVAL Row%, BYVAL Column%, Text\$, BYVAL TextColor%)

Where:

Text\$ is any text s

high-resolution gra Row% and Colun

LOCATE statemen

following formula

TextColors = fore

TextColor% holds

Syntax:

CALL SPrintOh

purpose:

GPrintOH prints a

screen.

Where:

LOCATE statement. Row% and Column% are the normal coordinates used by the BASIC

Text\$ is any text string.

TextColor% holds the combined foreground and background colors. The following formula can be used to set the colors used:

TextColor% = Foreground% + (Background * 256)

Comments:

operation. all programs or modules which use this routine will insure proper Many parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of

screen mode using this routine. The following example shows how to print a string to the CGA 2-color

'\$INCLUDE: 'GWDECL.BAS' SCREEN 2 DEFINT A-Z required for BYVAL's sets the monitor in CGA mode

with the Hercules gr The code fragment t

all programs or

maximum sp Many parar

Comments:

operation.

CALL GPrintOC2 (1, 10, "This text is on row 1", 1 + 0 * 256)

also allows you to specify a background color for that text string This routine is many times faster than the BASIC PRINT statement. It

See Also:

GPrint0VE, GPrint0H

See Also: GPrintOVE, GPrintOC

also allows for you to This routine is many

CALL SPrintOH (1, 1

CALL Screens

THE STATE ST

ON THIMHUS MASSES

amed n OMUR

TROOM NOON in

GPrint0H

subroutine contained in GW.LIB Assembler

Purpose:

GPrint0H prints a string on the 2-color high-resolution Hercules graphics

Syntax:

CALL GPrintOH (BYVAL Row%, BYVAL Column%, Text\$, BYVAL TextColor%)

used by the BASIC

Wal jestiolory

Row% and Column% are similar to the coordinates used by the BASIC LOCATE statement, but provide 90 columns instead of 80 in the Hercules high-resolution graphics screen.

Assembly Routines

Text\$ is any text string.

Found colors. The

TextColor% holds the combined foreground and background colors. The following formula can be used to set the colors used:

TextColor% = Foreground% + (Background * 256)

Comments:

t the beginning of ill insure proper

the CGA 2-color

le to provide the

Many parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper

The code fragment below shows how to print a string using inverse video with the Hercules graphics mode.

DEFINT A-Z

'required for BYVAL's 'sets the Hercules in graphics mode \$INCLUDE: 'GWDECL.BAS' CALL Screen3

CALL GPrintOH (1, 10, "This Text is on row 1", 0 + 1 * 256)

VT statement. It

at string.

This routine is many times faster than the BASIC PRINT statement. also allows for you to specify a background color for that text string.

See Also:

GPrint0VE, GPrint0C2

T SOFTWARE, BYC

GPrint0V256

Assembly Routines

subroutine contained in GW.LIB Assembler

purpose:

Purpose

GPrint0V256 prints a string on the 256-color VGA low-resolution graphics screen in a specified color.

Syntax

CALL GOVINTONE

graphics screens is GPrintOVE prints

Where:

CALL GPrint0V256 (BYVAL Row%, BYVAL Column%, Text\$, BYVAL TextColor%)

Assembly Routines

LOCATE statement in SCREEN 13. Row% and Column% are the normal coordinates used by the BASIC

Text\$ is any text string.

following formula can be used to set the colors used: TextColor% holds the combined foreground and background colors. The

following formula

TextColor% = For

TextColor% holds Texts is any text s LOCATE statemen Row% and Colur

TextColor% = Foreground% + (Background * 256)

Comments:

operation. all programs or modules which use this routine will insure proper Many parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of maximum speed.

The following example shows how to print a string to the VGA using the color blue for the foreground, and the color grey for the background.

The following

color blue for the fi

all program

maximum sp

'\$INCLUDE: 'GWDECL.BAS' SCREEN 13 DEFINT A-Z 'sets the monitor in VGA mode 'required for BYVAL's

GPrint0V256 1, 10, "This text is on row 1", 1 + 7 * 256

also allows you to specify a background color for that text string. This routine is many times faster than the BASIC PRINT statement.

See Also:

Print0C2, GPrint0H GPrint0VE, GPrint1VE, GPrint2VE, GPrint3V256, GPrint3VE,

See Also:

also allows you to sp This routine is many

@rint012 1, 10, "

SCREEN 12

330%

GPrintIVE, GPrint2

ON SHALLING METERSON

GPrint0VE

BUT CALLIS

subroutine contained in GW.LIB Assembler

Purpose:

GPrintOVE prints a string on the 16-color EGA and VGA high-resolution graphics screens in a specified color.

Syntax:

CALL GPrintOVE (BYVAL Row%, BYVAL Column%, Text\$, BYVAL TextColor%)

Where:

used by the BASIC

Row% and Column% are the normal coordinates used by the BASIC LOCATE statement.

Assembly Routines

Text\$ is any text string.

ground colors. The

TextColor% holds the combined foreground and background colors. The following formula can be used to set the colors used:

TextColor% = Foreground% + (Background * 256)

Comments:

it the beginning of

ue to provide the in insure proper he VGA using the

e background.

maximum speed. Including the file GWDECL BAS at the beginning of Many parameters for this routine are passed by value to provide the all programs or modules which use this routine will insure proper The following example shows how to print a string to the VGA using the color blue for the foreground, and the color grey for the background.

*SINCLUDE: 'GMDECL.BAS' 'required for BYVAL's CREEN 12 'sets the monitor in VGA mode

GPrintOVE 1, 10, "This text is on row 1", 1 + 7 * 256

This routine is many times faster than the BASIC PRINT statement. also allows you to specify a background color for that text string.

NT statement.

oxt String.

GPrint3VE, G-

GPrint1VE, GPrint2VE, GPrint3VE, GPrint0C2, GPrint0H

CRESCENT SOFTWARE, INC.

GPrint1VE

subroutine contained in GW.LIB Assembler

See Also:

GPrintOVE, GPrin

also allows for the

images without alt This routine is mi

Purpose:

of the background beneath the text. graphics screens in a specified color. This routine does not modify any GPrint1VE prints a string on the 16-color EGA and VGA high-resolution

Syntax:

CALL GPrint1VE (BYVAL Row%, BYVAL Column%, Text\$, BYVAL TextColor%)

Where:

Assembly Routines

Row% and Column% are the normal coordinates used by the BASIC LOCATE statement.

Text\$ is any text string.

following formula can be used to set the colors used: TextColor% holds the combined foreground and background colors. The

TextColor% = Foreground% + (Background * 256)

does not need to be placed in the above formula, but it will not matter if you leave it in the formula. GPrintXXX routines, in this case the background color is ignored and Although this is a standard formula for text colors with all of the

Comments:

operation. all programs or modules which use this routine will insure proper Many parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of

the background. color red that will write over any graphics objects, leaving the objects in The example below shows how to print a string to a VGA monitor in the

DEFINT A-Z

'GWDECL . BAS' 'required for BYVAL's

'sets the monitor in VGA mode

GPrint1VE 1, 10, "This text is on row 1", 4

ON BRIMAIN MARKED

Assembler Offills

Vod helestenluin thes not modify any

sed by the BASIC

round colors. The

s with all of the will not matter if or is ignored and

e to provide the Il insure proper

the beginning of

A monitor in the ng the objects in

also allows for the placement of text over even the most complex graphics This routine is many times faster than the BASIC PRINT statement. images without altering them.

See Also:

GPrint0VE, GPrint2VE, GPrint3VE

Assembly Routines

CRESCENT SOFTWARE, INC.

SOPTHARE INC

3 - 39

GPrint3V256

purpose:

GPrint2VE

subroutine contained in GW.LIB Assembler

Purpose:

of the 480 lines on the VGA display. graphics screens in a specified color. This routine allows text to be placed starting at any of the 350 lines on the EGA high-resolution display, or any GPrint2VE prints a string on the 16-color EGA and VGA high-resolution

CALL GPrint2VE (BYVAL Line%, BYVAL Column%, Text\$, BYVAL TextColor%)

w

Syntax: does not modify an at any of the 2001 screen in a specifie GPrint3V256 print

CALL Garintalizes

Line% and Colum Where:

Text\$ is any text st

values range from I mixed coordinate s

Where:

Line% and Column% specify the starting location of the string using the mixed coordinate system where column values range from 1 to 80 and line values range from 0 to 479 on a VGA monitor. This is the difference between this routine and GPrintOVE.

Text\$ is any text string.

following formula can be used to set the colors used: TextColor% holds the combined foreground and background colors. The

TextColor% = Foreground% + (Background * 256)

Comments:

or modules which use this routine will insure proper operation. speed. Including the file GWDECL.BAS at the beginning of all programs Parameters for this routine are passed by value to provide the maximum

color of white and a background color of red, at any Y coordinate on the The example below shows how to print a string of text, with a foreground VGA screen.

DEFINT A-Z
'\$INCLUDE: 'GWDECL.BAS' SCREEN 12 'required for BYVAL's

'sets the monitor in VGA mode

GPrint2VE 103, 10, "This text is on line 103", 15 + 4 * 256

also allows you to specify a background color for that text string. This routine is many times faster than the BASIC PRINT statement.

> printed at any one of on the screen leaving

SP-10t31256 142.6

SCREEN 13

The example below s

all programs or me Many parameters p

maximum speed. 1

Comments:

you leave it in

does not need to h

Although th

following form:

TextColor% holds

operation.

See Also:

GPrint0VE, GPrint1VE, GPrint3VE

ON THE LANGE MACE INC.

They in Chillips

GPrint3V256

Assembler subroutine contained in GW.LIB

Purpose:

VGA high-resolution sext to be placed ution display, or any

GPrint3V256 prints a string on the 256-color VGA low-resolution graphics screen in a specified color. This routine allows text to be placed starting at any of the 200 lines on the VGA low-resolution display. This routine does not modify any of the background beneath the text.

Syntax:

TYA TextColors)

CALL GPrint3V256 (BYVAL Line%, BYVAL Column%, Text\$, BYVAL TextColor%)

■ Where:

the string using the om 1 to 80 and line s is the difference

Line% and Column% specify the starting location of the string using the mixed coordinate system where column values range from 1 to 80 and line values range from 0 to 199.

Assembly Routines

Text\$ is any text string.

TextColor% holds the combined foreground and background colors. The following formula can be used to set the colors used:

ound colors. The

TextColor% = Foreground% + (Background * 256)

Although this is a standard formula for text colors with all of the does not need to be placed in the above formula, but it will not matter if GPrintXXX routines, in this case the background color is ignored and you leave it in the formula.

Comments:

g of all programs

ration.

de the maximum

coordinate on the

rith a foreground

Many parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation. The example below shows how to print a string of text over any objects The text can be on the screen leaving the objects in the background. printed at any one of the VGA's 200 line positions.

DEFINT A-Z
'\$INCLUDE: 'GWDECL.BAS' 'required for BYVA

.UDE: 'GWDECL.BAS' 'required for BYVAL's i 13 'sets the monitor in VGA mode

STIES.

952

GPrint3V256 102, 2, "This text is on line 102", 14

SOFTHARE, MC

GPrint3VE

purpose:

graphics screens in GPrint3VE prints a

images without altering them. also allows for the placement of text over even the most complex graphics This routine is many times faster than the BASIC PRINT statement.

See Also:

Assembly Routines

Where:

Syntax:

CALL GPrintSTE L

the background ber of the 480 lines on starting at any of th

GPrint0V256, GPrint0VE, GPrint1VE, GPrint2VE

■ 3 - 42

CRESCENT SOFTWARE, INC.

21 M33UN - 300 CM E, - 300 CM E, - 2-4 M1330

GPTHICIBE 302, 2,

on the screen leaving

all programs or mo operation. maximum speed. In: Many parameters fo

Comments: you leave it in the fo does not need to be

Although this is

GPrintXXX routine

following for

TextColor%

TextColor% holds

Text\$ is any text str between this routine values range from mixed coordinate sy Line% and Columi

printed at any one of t The example below st

CRESCENT SUFFINALE INC.

A September 1

GPrint3VE

Assembler subroutine contained in GW.LIB

Purpose:

GPrint3VE prints a string on the 16-color EGA and VGA high-resolution graphics screens in a specified color. This routine allows text to be placed starting at any of the 350 lines on the EGA high-resolution display, or any of the 480 lines on the VGA display. This routine does not modify any of the background beneath the text.

■ Syntax:

CALL GPrint3VE (BYVAL Line%, BYVAL Column%, Text\$, BYVAL TextColor%)

Assembly Routines

■ Where:

Line% and Column% specify the starting location of the string using the mixed coordinate system where column values range from 1 to 80 and line values range from 0 to 479 on a VGA monitor. This is the difference between this routine and GPrint1VE.

Text\$ is any text string.

TextColor% holds the combined foreground and background colors. The following formula can be used to set the colors used:

TextColor% = Foreground% + (Background * 256)

does not need to be placed in the above formula, but it will not matter if Although this is a standard formula for text colors with all of the GPrintXXX routines, in this case the background color is ignored and you leave it in the formula.

Comments:

maximum speed. Including the file GWDECL.BAS at the beginning of Many parameters for this routine are passed by value to provide the all programs or modules which use this routine will insure proper operation.

The example below shows how to print a string of text over any objects on the screen leaving the objects in the background. printed at any one of the VGA's 480 line positions.

DEFINT A-Z

**INCLUDE: 'GWDECL.BAS' 'required for BYVAL's
SCREEN 12 'sets the monitor in VGA mode

GPrint3VE 302, 2, "This text is on line 302", 14

SOFTWARE, INC.

LineBF2VE

purpose:

images without altering them. also allows for the placement of text over even the most complex graphics This routine is many times faster than the BASIC PRINT statement.

See Also:

GPrintOVE, GPrint1VE, GPrint2VE

Syntax:

CALL LineBrz/E (8 BYWAL LineColor

operations for draw than LineBFVE.

Unlike LineBFVE t LineBF2VE draws a

Where:

The coordinate pair

of the screen.

LineColor% is the

color value shown in

6 H3380S 300TOHIT, 7-Y INIAGO

To create a filled bo

Assembly Routines

CRESCENT SOFTWARE, INC.

CHESCENT SUPPLYAVE DVC

See Also:

or modules which use speed. Including the

All parameters for this Comments;

CALL LineBF2NE

LineVE, LineBFVE,

LineBF2VE

Assembler subroutine contained in GW.LIB

Purpose:

LineBF2VE draws a filled box on an EGA or VGA high-resolution screen. Unlike LineBFVE this routine cannot utilize OR, AND, XOR, and PSET operations for drawing the filled box. This routine is however 40% faster than LineBFVE.

■ Syntax:

CALL LineBF2VE (BYVAL x1%, BYVAL y1%, BYVAL x2%, BYVAL y2%, BYVAL LineColor%)

Assembly Routines

■ Where:

The coordinate pairs (x1%, y1%) and (x2%, y2%) are within the range of the screen.

LineColor% is the color of the line.

To create a filled box in the color grey, you would pass LineBF2VE the color value shown in the following example.

DEFINT A-Z
*SINCLUDE: 'GWDECL.BAS' 'required for BYVAL's

INCLUDE: 'GWDECL.BAS' 'required for BYYAL's
REK 9 'sets the monitor in an EGA mode

LineColor% = 7 CALL LineBF2VE (100, 100, 200, 200, LineColor%)

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

See Also:

LineVE, LineBFVE, LineBVE

LineBFVE

Assembler subroutine contained in GW.LIB

Purpose:

the filled box. This routine can utilize OR, AND, XOR, and PSET operations for drawing LineBFVE draws a filled box on an EGA or VGA high-resolution screen.

■ Syntax:

CALL LineBFVE (BYVAL x1%, BYVAL y1%, BYVAL x2%, BYVAL y2%, BYVAL LineColor%)

■ Where:

Assembly Routines

of the screen The coordinate pairs (x1%, y1%) and (x2%, y2%) are within the range

Where:

The coordinate pair of the screen.

Syntax:

CALL LineSHE (8Y) BYVAL LineColor LineBVE draws a b This routine can uti

the box.

box. table to change the logical operation performed when drawing the filled LineColor% is the color of the line. Add one of the values below in the

table to change the

LineColor% is the

VALUE	LOGICAL OPERATION
0	PSET (Replace)
2048	OR
4096	AND
6144	XOR

Any other values will produce unusual (and probably undesirable) effects.

LineBFVE the color value shown in the following example. To create a rubber-banding effect with the color grey, you would pass

LineBVE the color v

Any other values wi

SCREEN 9 'sets the monitor in an EGA mode LineColor% = 7 + 6144

CALL LineBFVE (100, 100, 200, 200, LineColor%)

Comments:

or modules which use this routine will insure proper operation speed. Including the file GWDECL BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

See Also:

LineVE, LineBF2VE, LineBVE

■ See Also:
LineVE, LineBFVE

speed. Including the fi

Comments; All parameters for this

CALL Lineste (10)

LineColork = 7 + 6

OBSCENT SUFFINALE INC.

LineBVE

Arred n QV, LIB

ph-resolution street Perations for drawing

Assembler subroutine contained in GW.LIB

Purpose:

This routine can utilize OR, AND, XOR, and PSET operations for drawing LineBVE draws a box outline on an EGA or VGA high-resolution screen. the box.

Syntax:

CALL LineBUE (BYVAL x1%, BYVAL y1%, BYVAL x2%, BYVAL y2%, BYVAL LineColor%)

Where:

are within the range

values below in the drawing the filled

The coordinate pairs (x1%, y1%) and (x2%, y2%) are within the range of the screen.

Assembly Routines

LineColor% is the color of the line. Add one of the values below in the table to change the logical operation performed when drawing the line.

LOGICAL OPERATION	PSET (Replace)	OR	AND	XOR
VALUE	0	2048	4096	6144

Any other values will produce unusual (and probably undesirable) effects.

To create a rubber-banding effect with the color grey, you would pass LineBVE the color value shown in the following example.

, you would pass

desirable) effects.

*INCLUDE: 'GWDECL.BAS' DEFINT A-Z

'sets the monitor in an EGA mode 'required for BYVAL's

LineColor% = 7 + 6144 CALL LineBVE (100, 100, 200, 200, LineColor%)

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

See Also:

ide the maximum

of all programs

Tation.

LineVE, LineBFVE

T SOFTWARE, INC.

LineStepVE

Assembler subroutine contained in GW.LIB

See Also: Line VE, Line ToSt

All parameters for speed. Including to speed, which or modules which

Purpose:

step values from the last point drawn to. This routine can utilize OR, AND, XOR, and PSET operations for drawing the line. LineStepVE draws a line on an EGA or VGA high-resolution screen using

Syntax:

CALL LineStepVE (BYVAL StepX1%, BYVAL StepY1%, BYVAL StepX2%, BYVAL_ StepY2%, BYVAL LineColor%)

Where:

Assembly Routines

are within the range of the screen and are distances from the last point The coordinate pairs (StepX1%, StepY1%) and (StepX2%, StepY2%)

table to change the logical operation performed when drawing the line. LineColor% is the color of the line. Add one of the values below in the

VALUE	LOGICAL OPERATION
0	PSET (Replace)
2048	OR
4096	AND
6144	XOR

Any other values will produce unusual (and probably undesirable) effects.

LineStepVE the color value shown in the following example. Note that the line will be drawn from coordinate (100, 100) to (200, 200). To create a rubber-banding effect with the color grey, you would pass

DEFINT A-Z

'SINCLUDE: 'GWDECL.BAS' 'required for BYVAL'S

SCREEN 9

LineColor% = 7 + 614

CALL SetLastCoord(50, 50)

CALL LineStepVE (50, 50, 150, LineColor%)

CHARGE MACHINE THE

Assembly Routines

Comments:

ared n CALIB

utine ca unite OR

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

See Also:

LineVE, LineToStepVE

Stepitzi, STIM

Assembly Routines

epX2%, StepY2%) from the last point

Values below in the

drawing the line.

ample. Note that

apou

idesirable) effects.

CRESCENT SOFTWARE, INC.

T SOFTWARE INC

3 - 49

LineToVE

subroutine contained in GW.LIB Assembler

Purpose:

from the last cour Line To Step VE dra

This routine can ut

the line.

Purpose:

the line. the last coordinate drawn to, to the coordinate specified to this routine. This routine can utilize OR, AND, XOR, and PSET operations for drawing LineToVE draws a line on an EGA or VGA high-resolution screen from

Syntax:

CALL LineToster

Where:

Syntax:

CALL LineToVE (BYVAL ToX%, BYVAL ToY%, BYVAL LineColor%)

Where:

Assembly Routines

The coordinate pair (ToX%, ToY%) is within the range of the screen.

table to change the logical operation performed when drawing the line. LineColor% is the color of the line. Add one of the values below in the

table to change the

LineColor% is the and specifies a dist The coordinate pair

10.00	4096
	AND

Any other values will produce unusual (and probably undesirable) effects.

LineToVE the color value shown in the following example. To create a rubber-banding effect with the color grey, you would pass

LineToStep VE the co

To create a rubber-Any other values

```
CALL SetLastCoord(100, 100)
                                              LineColor% = 7 + 6144
                                                                                             SCREEN 9
                                                                                                                                         DEFINT A-Z
CALL LineToVE (200, 200, LineColor%)
                                                                                                               '$INCLUDE: 'GWDECL.BAS'
                                                                                      'sets the monitor in an EGA mode
                                                                                                             'required for BYVAL's
```

Comments:

or modules which use this routine will insure proper operation speed. Including the file GWDECL.BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

Comments:

CALL LineToStepiE CALL SetLastCoord

LineColors = 7 + 61

STREEN 9 SOUTER

See Also:

LineVE, LineToStepVE

THE THE PASSED IN THE PASSED IN CO.

See Also:LineVE, Lin or modules which use speed. Including the fi All parameters for this

LineToStepVE

aired in OVIIB

subroutine contained in GW.LIB Assembler

Purpose:

Solve in screen from ified to this routine. Perations for drawing

Line To Step VE draws a line on an EGA or VGA high-resolution screen This routine can utilize OR, AND, XOR, and PSET operations for drawing from the last coordinate drawn to, to the point a specified distance away. the line.

Syntax:

CALL LineToStepVE (BYVAL StepX%, BYVAL StepY%, BYVAL LineColor%)

Where:

values below in the drawing the line.

ge of the streen

The coordinate pair (StepX%, StepY%) is within the range of the screen and specifies a distance from the last point drawn to.

Assembly Routines

LineColor% is the color of the line. Add one of the values below in the table to change the logical operation performed when drawing the line.

VALUE LC 0 PS 2048 OF 4096 AN	LOGICAL OPERATION	PSET (Replace)	~	Ą	8
VALUE 0 2048 4096	LOC	PSE	OR	AND	XOR
	VALUE	0	2048	4096	6144

Any other values will produce unusual (and probably undesirable) effects.

desirable) effects. you would pass

To create a rubber-banding effect with the color grey, you would pass LineToStepVE the color value shown in the following example.

*INCLUDE: 'GWDECL.BAS' SCREEN 9

'required for BYVAL's 'sets the monitor in an EGA mode

LineColor% = 7 + 6144 CALL SetLastCoord (100, 100) CALL LineToStepVE (100, 100, LineColor%)

Comments:

ide the maximum g of all programs

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL. BAS at the beginning of all programs or modules which use this routine will insure proper operation.

See Also:LineVE, LineStepVE, LineToStepVE

T SOFTWARE INC.

LineVE

subroutine contained in GW.LIB Assembler

purpose:

end of your progr ScreenO sets a mor

routines in Graphi

Purpose:

routine can utilize OR, AND, XOR, and PSET operations for drawing the LineVE draws a line on an EGA or VGA high-resolution screen. . This

Syntax:

CALL LineVE (BYVAL x1%, BYVAL y1%, BYVAL x2%, BYVAL y2%, BYVAL LineColor%)

Where:

Syntax:

CALL ScreenO(Kee

Where:

Assembly Routines

of the screen. The coordinate pairs (x1%, y1%) and (x2%, y2%) are within the range

table to change the logical operation performed when drawing the line LineColor% is the color of the line. Add one of the values below in the

> return to it. A valu the information on to clear the screen KeepData% is a tr

BASIC does not he

mode is set. Comments:

XOR	6144
AND	4096
OR	2048
PSET (Replace)	0
LOGICAL OPERATION	VALUE

monitor into of the Graphics

REFINI A-2

The following ex

Any other values will produce unusual (and probably undesirable) effects.

LineVE the color value shown in the following example To create a rubber-banding effect with the color grey, you would pass

CALL Screen((T)

```
CALL LineVE (100, 100, 200, 200, LineColor%)
                           LineColor% = 7 + 6144
                                                                         SCREEN 9
                                                                                                                     DEFINT A-Z
                                                                                                '$INCLUDE: 'GWDECL.BAS'
                                                                                         'required for BYVAL's
                                                                  'sets the monitor in an EGA mode
```

Comments:

or modules which use this routine will insure proper operation speed. Including the file GWDECL.BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

See Also:

DEMOSCRN.BAS. To take advantage of makes this work the this screen mode. A need to understand a If you're going to u

Screen13 Screen!, Screen?,

See Also:

LineToVE, LineToStepVE CircleVE, DrawPointVE, LineBFVE, LineBVE, LineStepVE,

OROBIT STATISTICS

aned novine

Mution Screen. This isomethic the

Assembler subroutine contained in GW.LIB

Purpose:

Screen0 sets a monitor into text mode. This routine should be used at the end of your programs if you have used one of the other screen setting routines in Graphics Workshop.

Syntax:

CALL ScreenO(KeepData%)

■ Where:

re within the range

values below in the

drawing the line.

to clear the screen when setting the screen. This gives you an ability BASIC does not have. If you set this value to anything other than 0, all KeepData% is a true or false value which tells the routine whether or not the information on the screen will remain in memory and you can even return to it. A value of 0 will clear the screen at the same time the video mode is set.

Assembly Routines

Comments:

of the Graphics Workshop Screen routines which had previously set the The following example sets the monitor into text mode after using another monitor into the VGA graphics mode.

DEFINT A-Z

AS' 'required for BYVAL's 'sets the monitor in VGA mode

'\$INCLUDE: 'GWDECL.BAS' 'required for CALL Screen12(0) 'sets the mon

CALL Screen0(0) 'sets the

you would pass

'sets the monitor into text mode

If you're going to utilize the KeepData feature of this routine, you will need to understand a little about how the video memory is structured for this screen mode. Most programmers will pass only a value of 0 which makes this work the same as the BASIC SCREEN statement equivalent. To take advantage of the KeepData feature, examine the example program DEMOSCRN.BAS.

■ See Also:

de the maximum of all programs LineStep VE.

Screen2, Screen3, Screen7, Screen9, Screen11, Screen12, Screen1, Screen13

SOFTWARE, D.C.

subroutine contained in GW.LIB Assembler

Purpose:

Screen2 sets an Co

This is a replaceme

CALL Screen2/Kee

Purpose:

This is a replacement for the SCREEN 1 statement in BASIC Screen1 sets an CGA monitor into the 320 by 200 pixel 4-color mode.

CALL Screen1(KeepData%)

Where:

Assembly Routines

mode is set. return to it. A value of 0 will clear the screen at the same time the video the information on the screen will remain in memory and you can even BASIC does not have. If you set this value to anything other than 0, all KeepData% is a true or false value which tells the routine whether or not to clear the screen when setting the screen. This gives you an ability

> return to it. A value the information on BASIC does not ha to clear the screen KeepData% is a tri Where:

mode is set.

Comments:

mode, and clears The following exam

DEFINT A-Z

Comments:

mode, and clears the screen at the same time. The following example sets the monitor into the CGA 4-color graphics

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS'

CALL Screen1(0)

'required for BYVAL's

'sets the monitor in CGA mode

To take advantage of the KeepData feature, examine the example program DEMOSCRN.BAS. this screen mode. Most programmers will pass only a value of 0 which makes this work the same as the BASIC SCREEN statement equivalent. If you're going to utilize the KeepData feature of this routine, you will need to understand a little about how the video memory is structured for

> makes this work the need to understand a If you're going to

uns screen mode. N

graphics mode. Screen0. To return to text mode at the completion of your program, use the routine Otherwise your program will return to DOS while still in a

See Also:

Screen0, other screen setting routines

See Also:

graphics mode. Screeno. Otherwise To return to text mode DEMOSCRN.BAS. To take advantage of t

Screent, other screen

ON BRYMAIS MANNED

red in CMUB

Assembler subroutine contained in GW.LIB

Purpose:

Pixel 4-color mode.

BASIC

Screen2 sets an CGA monitor into the 640 by 200 pixel 2-color mode. This is a replacement for the SCREEN 2 statement in BASIC.

Syntax:

Screen2(KeepData%)

Where:

utine whether or not ives you an ability ng other than 0, ail ane time the video y and you can even

KeepData% is a true or false value which tells the routine whether or not the information on the screen will remain in memory and you can even to clear the screen when setting the screen. This gives you an ability BASIC does not have. If you set this value to anything other than 0, all return to it. A value of 0 will clear the screen at the same time the video mode is set.

Assembly Routines

Comments:

4 4-color graphics

The following example sets the monitor into the CGA 2-color graphics mode, and clears the screen at the same time.

DEFINT A-Z

\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's

CALL Screen2(0)

'sets the monitor in CGA mode

this screen mode. Most programmers will pass only a value of 0 which If you're going to utilize the KeepData feature of this routine, you will need to understand a little about how the video memory is structured for makes this work the same as the BASIC SCREEN statement equivalent. To take advantage of the KeepData feature, examine the example program DEMOSCRN.BAS. To return to text mode at the completion of your program, use the routine Otherwise your program will return to DOS while still in a graphics mode.

See Also:

a, use the routine

while still in a

ement equivalent.

routine, you will Is structured for value of 0 which xample program Screen0, other screen setting routines

subroutine contained in GW.LIB Assembler

Purpose:

This is a replaceme Screen 7 sets an EC Screen.

Syntax:

CALL Screen/Ikee

Where:

Purpose

Screen3 sets a Hercules monitor into the 720 by 350 pixel 2-color mode. This is a replacement for the SCREEN 3 statement in BASIC.

Syntax:

CALL Screen3

Comments:

Assembly Routines

then this routine will serve your Hercules graphics needs without requiring all of your Hercules graphics needs using the BASIC POKE statement, MSHERC.COM for simple Hercules support. If you are using the PCXArrayH or DispPCXH routines to display a PCX file, or can achieve MSHERC.COM to be loaded. If you use this routine to set the screen mode you will NOT need to run

> return to it. A value the information on t BASIC does not have to clear the screen KeepData% is a tru

mode is set. Comments:

mode, and clears th The following exact

screen at the same time. utility. without the need of the MSHERC.COM Terminate and Stay Resident The following example shows how to turn on the Hercules graphics mode Specifying not to keep the data will tell the routine to clear the

DEFINT A-Z

'\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's

CALL Screen3(0)

'sets the Hercules into graphics mode

Of Buros as mol m

is screen mode. M

MSHERC.COM, take a look at the Graphics Workshop function Herc-If you need more of the features BASIC provides using the There% which determines whether or not the TSR is loaded in memory.

graphics mode. Screen0. To return to text mode at the completion of your program, use the routine Otherwise your program will return to DOS while still in a

See Also:

HercThere%, MSHERC.COM, Screen0

See Also:

graphics mode. Screen). Otherwise y To return to text mode

DEMOSCRN.BAS. To take advantage of the makes this work the s need to understand a

Screen(), other screen s

ined in CMUB

pixel 2-color mode

in BASIC

subroutine contained in GW.LIB Assembler

Purpose:

Screen7 sets an EGA monitor into the 640 by 350 pixel 2-color mode. This is a replacement for the SCREEN 7 statement in BASIC.

Syntax:

CALL Screen7(KeepData%)

Where:

Il NOT need to nun you are using the file, or can achieve S without requiring POKE Statement,

KeepData% is a true or false value which tells the routine whether or not to clear the screen when setting the screen. This gives you an ability BASIC does not have. If you set this value to anything other than 0, all the information on the screen will remain in memory and you can even return to it. A value of 0 will clear the screen at the same time the video mode is set.

Comments:

iles graphics mode and Stay Resident outine to clear the

The following example sets the EGA monitor into a monochrome graphics mode, and clears the screen.

'required for BYVAL's 'sets the monitor in EGA mono 'GWDECL.BAS' CALL Screen7(0) DEFINT A-Z \$INCLUDE:

If you're going to utilize the KeepData feature of this routine, you will need to understand a little about how the video memory is structured for this screen mode. Most programmers will pass only a value of 0 which makes this work the same as the BASIC SCREEN statement equivalent. To take advantage of the KeepData feature, examine the example program DEMOSCRN.BAS.

using the TSR

ics mode

p function Hercaded in memory. a, use the routine S while still in a

To return to text mode at the completion of your program, use the routine Screen0. Otherwise your program will return to DOS while still in a graphics mode.

See Also:

Screen0, other screen setting routines

Assembly Routines

subroutine contained in GW.LIB Assembler

purpose;

Screen I I sets a W

This is a replacement

ScreenTI

Purpose:

Screen9 sets an EGA monitor into the 640 by 350 pixel 16-color mode. This is a replacement for the SCREEN 9 statement in BASIC.

Syntax:

CALL Screen9(KeepData%)

Where:

Syntax:

CALL Scree

BASIC does not ha KeepData% is a tr

return to it. A valu the information on to clear the screen

Where:

the information on the screen will remain in memory and you can even mode is set. return to it. A value of 0 will clear the screen at the same time the video BASIC does not have. If you set this value to anything other than 0, all to clear the screen when setting the screen. KeepData% is a true or false value which tells the routine whether or not This gives you an ability

Comments:

screen mode, and clear the screen at the same time. The example below shows how to set the EGA into its high-resolution

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's

CALL Screen9(0) 'sets the monitor in EGA mode

makes this work the same as the BASIC SCREEN statement equivalent. this screen mode. Most programmers will pass only a value of 0 which need to understand a little about how the video memory is structured for If you're going to utilize the KeepData feature of this routine, you will DEMOSCRN.BAS. To take advantage of the KeepData feature, examine the example program

graphics mode. Screen0. To return to text mode at the completion of your program, use the routine Otherwise your program will return to DOS while still in a

See Also:

Screen0, other screen setting routines

See Also: Streen, other streen

graphics mode, Screen). Otherwise To return to text mode DEMOSCRN.BAS.

To take advantage of t makes this work the s

If you're going to ut this screen mode. M need to understand a

THE Screen!

screen at the sa

FW MILESO

The brief code from Comments: mode is set.

DEN JANUARON MADORED

a ed in CMLIB

pixel 16-color mode.

in BASIC

subroutine contained in GW.LIB Assembler

Purpose:

Screen11 sets a VGA monitor into the 640 by 480 pixel 2-color mode. This is a replacement for the SCREEN 11 statement in BASIC.

Syntax:

CALL Screen11(KeepData%)

Where:

utine whether or not gives you an ability ing other than 0, all y and you can even same time the video

to clear the screen when setting the screen. This gives you an ability BASIC does not have. If you set this value to anything other than 0, all the information on the screen will remain in memory and you can even KeepData% is a true or false value which tells the routine whether or not return to it. A value of 0 will clear the screen at the same time the video mode is set.

Comments:

its high-resolution

high-resolution monochrome mode. The call to Screen11 will clear the The brief code fragment below shows how to set the VGA monitor into a screen at the same time.

'GWDECL.BAS' \$INCLUDE:

'required for BYVAL's

CALL Screen11(0)

y is structured for value of 0 which example program

'sets the monitor in VGA mono

If you're going to utilize the KeepData feature of this routine, you will need to understand a little about how the video memory is structured for this screen mode. Most programmers will pass only a value of 0 which makes this work the same as the BASIC SCREEN statement equivalent. To take advantage of the KeepData feature, examine the example program DEMOSCRN.BAS. To return to text mode at the completion of your program, use the routine Otherwise your program will return to DOS while still in a graphics mode.

See Also:

Screen0, other screen setting routines

Assembly Routines

n, use the routine S while still in a

Screen12

Assembly Routines

subroutine contained in GW.LIB Assembler

Purpose:

Screen12 sets a VGA monitor into the 640 by 480 pixel 16-color mode This is a replacement for the SCREEN 12 statement in BASIC

Syntax:

CALL Screen12(KeepData%)

Where:

KeepData % is a tri

Syntax:

CALL Screen131

This is a replaceme Screen 13 sets a W

Assembly Routines

return to it. A value of 0 will clear the screen at the same time the video the information on the screen will remain in memory and you can even KeepData% is a true or false value which tells the routine whether or not to clear the screen when setting the screen. This gives you an ability mode is set. BASIC does not have. If you set this value to anything other than 0, all

> return to it. A value the information on

Comments: mode is set.

The following exam

BASIC does not have to clear the screen

Comments:

and clears the screen at the same time. The following example sets the VGA into its high-resolution color mode,

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's

CALL Screen12(0) 'sets the monitor in VGA mode

makes this work the same as the BASIC SCREEN statement equivalent. this screen mode. Most programmers will pass only a value of 0 which need to understand a little about how the video memory is structured for If you're going to utilize the KeepData feature of this routine, you will DEMOSCRN.BAS. To take advantage of the KeepData feature, examine the example program

nakes this work the

graphics mode. Screen0. To return to text mode at the completion of your program, use the routine Otherwise your program will return to DOS while still in a

See Also:

Screen0, other screen setting routines

See Also; Screen(), other screen

graphics mode. Screen). Otherwise To return to text mode DEMOSCRN.BAS. To take advantage of t this screen mode. M need to understand a If you're going to ut

CHI BRYMAN MARKETT

Aned in CM1/B Assembler

Dixel 16-color mode.

In BASIC

subroutine contained in GW.LIB Assembler

Purpose:

Screen13 sets a VGA monitor into the 320 by 200 pixel 256-color mode. This is a replacement for the SCREEN 13 statement in BASIC.

Syntax:

CALL Screen13(KeepData%)

utine whether or not

Y and you can even ives you as ability ng other than 0, all ame time the video

KeepData% is a true or false value which tells the routine whether or not This gives you an ability BASIC does not have. If you set this value to anything other than 0, all the information on the screen will remain in memory and you can even return to it. A value of 0 will clear the screen at the same time the video to clear the screen when setting the screen. mode is set.

Comments:

lution color mode.

The following example sets the VGA monitor into its 256-color mode, and clears the screen at the same time.

'GWDECL.BAS' .\$INCLUDE:

'required for BYVAL's CALL Screen13(0)

If you're going to utilize the KeepData feature of this routine, you will need to understand a little about how the video memory is structured for this screen mode. Most programmers will pass only a value of 0 which makes this work the same as the BASIC SCREEN statement equivalent. To take advantage of the KeepData feature, examine the example program 'sets the monitor in 256-color mode DEMOSCRN.BAS. To return to text mode at the completion of your program, use the routine Screen0. Otherwise your program will return to DOS while still in a graphics mode.

See Also:

Screen0, other screen setting routines

Assembly Routines

routine, you will example program is structured for value of 0 which

1, use the routine

while still in a

CRESCENT SOFTWARE, INC.

3-61

ClearScreenA

п Ригрозе:

Syntax:

CALL ClearScreen

by the various GPr The routine Clears

value at position 1, The example below

DEFINIT A-Z "\$INCLUDE: SCREEN 12

3000 3000

Comment

■ See Also: GetCharacter

CALL ClearScreen

Chark = GetCharact

Section 3: General Screen Manipulation Routines

screen to be scrolled in any direction. The routines in this section are routines for which there is no BASIC equivalent. Some of the routines move regions of the graphics screen to other locations. The GScrollVE routine will allow any region of a graphics

Assembly Routines

■ 3-62

CRESCENT SOFTWARE, INC.

ON SPRINGLING LINES

ClearScreenArray

Assembler subroutine contained in GW.LIB

Purpose:

Dere is no BASIC September Screen in September September

utines

The routine ClearScreenArray is used to reset the screen array maintained by the various GPrint??? routines.

Syntax:

CALL ClearScreenArray

Comment:

The example below resets the screen array and then checks the character value at position 1,1.

Assembly Routines

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' SCREEN 12

'required for BYVAL's 'sets the monitor in VGA mode

CALL ClearScreenArray

Char% = GetCharacter%(1, 1)
'Char% will equal 32 (a space) at this point.

See Also:

GetCharacter%

DrawByteVE

purpose:

DrawByteVE will di

physical location in

ClearVE

subroutine contained in GW.LIB Assembler

Purpose:

state. The routine ClearVE will reset the EGA and VGA registers to a known

Syntax:

CALL ClearVE

Comment:

The example below clears the VGA registers.

Assembly Routines

DEFINT A-Z
'\$INCLUDE: 'GWDECL.BAS'
SCREEN 12 required for BYVAL's sets the monitor in VGA mode

Where:

■ Syntax:

CALL DrawByteVE(BY BYWAL ByteColors

pixels at a time. bit pattern will be m

CALL ClearVE

routine can be used to reset all of them to a known state. registers on your own. If you do modify the EGA or VGA registers, this There may be no need for this routine unless you modify the EGA or VGA

will be colored on the BitPattern% is a byte address if you need it ScreenPosition % is

function MakeAddre

CARCELL PARTHARE INC.

CAL Drang/tels (Addr

DitPatterny = 255

the color red The following example or modules which use

SCREEN 12 SCREEN 12 SCREEN 12

speed. Including the fi All parameters for this Comments;

specified by th

ByteColor% is a

DrawByteVE

ained in CM IIB

registers to a known

Assembler subroutine contained in GW.LIB

Purpose:

physical location in the specified color. Only those bits specified in the DrawByteVE will draw a byte to an EGA or VGA screen at the specified bit pattern will be modified. Using this routine allows you to plot up to 8 pixels at a time.

■ Syntax:

CALL DrawByteVE(BYVAL ScreenPosition%, BYVAL BitPattern%, BYVAL ByteColor%)

Assembly Routines

■ Where

ScreenPosition% is an actual memory address for the EGA screen. The function MakeAddressVE% can provide you with the EGA memory address if you need it.

> y the EGA or VGA VGA registers, this

BitPattern% is a byte value for which every bit set in its binary equivalent will be colored on the screen. ByteColor% is a color from 0 to 15 for the bits to be set. Only those bits specified by the BitPattern% parameter will receive the color in Byte-

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The following example sets the first eight horizontal bits on the screen to the color red.

DEFINT A-Z
'\$IMCLUDE: 'GWDECL.BAS' 'required for BYVAL's
SCREEN 12 'sets the monitor in VGA mode

Address% = 0 'first address of the screen BitPattern% = 255 'set all bits

CALL DrawByteVE (Address%, BitPattern%, 4)

SOFTWARE, INC.

DrawbyteVEC

Purpose:

DrawByte VEOpts 1

specified physical lo in the bit pattern wi

The DrawByteVE routine provides the second fastest drawing speed to the screen. The fastest drawing speed can be obtained by using the LineVE statement.

See Also:

DrawPointVE, LineVE, MakeAddressVE%

Assembly Routines

Where:

function MakeAddro

BitPattern% is about address if you need it Screen Position % is

will be colored on t

ByteColor%

syntax:

CALL OrandytevEOp BYVAL ByteColor

operators OR, AND up to 8 pixels at a tir

■ 3-66

CRESCENT SOFTWARE, INC.

John Britans

The following examples with the first eight

All parameters for this repeat. Including the fil or modules which use t

Comments: Any other values will p

/£

2048

logical operator will specified by the Bird Color%. If you add o

Street, Street, the

DrawByteVEOpts

Assembler subroutine contained in GW.LIB

Purpose:

DrawByteVEOpts will draw a byte to an EGA or VGA screen at the specified physical location in the specified color. Only those bits specified in the bit pattern will be modified. Using this routine allows you to plot up to 8 pixels at a time. In addition, this routine can use any of the logical operators OR, AND, or XOR.

■ Syntax:

CALL DrawByteVEOpts(BYVAL ScreenPosition%, BYVAL BitPattern%, BYVAL ByteColor%)

Assembly Routines

Where.

function MakeAddressVE% can provide you with the EGA memory ScreenPosition% is an actual memory address for the EGA screen. address if you need it. BitPattern% is a byte value for which every bit set in its binary equivalent will be colored on the screen. ByteColor% is a color from 0 to 15 for the bits to be set. Only those bits specified by the BitPattern% parameter will receive the color in Byte-Color %. If you add one of the below values to the color, then the associated logical operator will be performed.

VALUE 0 2048 4096	LOGICAL OPERATION PSET (Replace) OR AND
6144	XOR

Any other values will produce unusual (and probably undesirable) effects.

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL. BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The following example uses to logical operation OR, to combine the color blue with the first eight horizontal pixels on the screen.

```
DEFINT A-Z
'$INCLUDE: 'GWDECL.BAS'
SCREEN 12
required for BYVAL's sets the monitor in VGA mode
```

Address% = 0 'first address on the screen BitPattern% = 255 'set all bits CALL DrawByteVEOpts (Address%, BitPattern%, 1 + 2048)

LineVE statement. to the screen. The fastest drawing speed can be obtained by using the The DrawByteVEOpts routine provides the second fastest drawing speed

| Syntax:

CALL Fade2EGA[BIV

Where:

will range between I wide by 8 pixels high The video screen is

all five patterns on or to one of five pre-con The variable Randor

All parameters for speed. Includi

Comments: region.

or modules whi

purpose:

video page to anoth Fade2EGA uses a ra

See Also:

DrawByteVE, DrawPointVE, LineVE, MakeAddressVE%

Assembly Routines

CHROSELL SURFLANDE DAG

The sequence is prequently of the property of

ESIL FAGES III. 1

Some sample graphics LINE (0, 0) - (859, 3 LINE (5, 5) - (100, 4 CIRCLE (30, 31), 27, PODPY 0, 1

to transfer the image. page, and then uses th The below example dr

SCREEN 9
SCREEN 9

A POLE

Fade2EGA

Assembler subroutine contained in GW.LIB

Purpose:

fastest drewing speed obtained by wing the

Fade2EGA uses a random sequence of points to transfer images from one video page to another. This routine is meant for use with EGA adapters.

■ Syntax:

SVEG

CALL Fade2EGA(BYVAL GridX%, BYVAL GridY%, BYVAL RandomPortion%)

■ Where:

The video screen is broken up into a grid where each region is 40 pixels wide by 8 pixels high. GridX% is a number between 1 and 16. GridY% will range between 1 and 44.

Assembly Routines

The variable RandomPortion% is a number between 1 and 5. It refers to one of five pre-configured random patterns for the grid region. Placing all five patterns on one grid region will create a solid image on that grid

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL. BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The below example draws some images, copies them to the second video page, and then uses the Fade2EGA routine on the upper-left grid region to transfer the image.

```
DEFINIT A-Z

'$INCLUDE: 'GADECL.BAS' 'required for BYVAL's

SCREN 9 'sets the monitor in EGA mode
'some sample graphics

LINE (0, 0) - (639, 349), 1, B

LINE (5, 5) - (100, 40), 2, BF

CIRCLE (5, 5) - (100, 40), 2, BF

CIRCLE (30, 30), 4, 4

PODPY 0, 1

CLS

CLS

FOR T = 1 TO 5

CALL Fade2EGA (1, 1, T)
```

The sequence is pseudo random and is specified by the data in FADEDATZ.GW.

T SOFTWARE, INC.

When a grid region is specified and a RandomPortion% is specified, it will transfer one-fifth of the screen image. Each grid region must be specified a total of five times, each time with a different random pattern specified to transfer the entire image to that region. The order for the random patterns is not important, as long as every region receives each RandomPortion at least once.

purpose:

FadeEGA uses a rain

video page to anoth

when there is a large portion of the screen which will not be changed by this routine, the effect is much more pleasing to the eye and doesn't appear background of the two screen images is going to be similar. It is advisable to use this routine and this effect when the overall Any time

Where:

CALL FadeEGA(BYFAL

example for this routine. transfer a screen from the background, and should be utilized as an The BASIC subroutines NightFall and RandomFade use this routine to

See Also:

FadeEGA, NightFall, RandomFade

all ten patterns on or to one of ten pre-cont The variable Randon will range between I wide by 32 pixels hig The video screen is t

Comments:

All parameters i

ON SHIPLING MANNERS

FADEDATA, GW. The sequence is pseu

F00 T = 1 TO 10

OUL FACEGO (L.

transfer the image. The below example dr

74 M 130

page, and then uses th

or modules whi. speed. Includi

SCREEN 9

A MARIA SERVICE SERVIC

FadeEGA

Assembler Subroutine contained in GW.LIB

Purpose:

et when the overall into the changed by time eard doesn't appear

FadeEGA uses a random sequence of points to transfer images from one video page to another. This routine is meant for use with EGA adapters.

■ Syntax:

CALL FadeEGA(BYVAL GridX%, BYVAL GridY%, BYVAL RandomPortion%)

■ Where:

e use this noutine to d be utilized as an

The video screen is broken up into a grid where each region is 160 pixels wide by 32 pixels high. GridX% is a number between I and 4. GridY% will range between 1 and 11.

Assembly Routines

The variable RandomPortion% is a number between 1 and 10. It refers to one of ten pre-configured random patterns for the grid region. Placing all ten patterns on one grid region will create a solid image on that grid

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation. The below example draws some images, copies them to the second video page, and then uses the FadeEGA routine on the upper-left grid region to transfer the image.

```
DEFINI A-Z
SINCLUDE: 'GADECL.BAS' 'required for BYVAL's
SCREEN 9 'sets the monitor in EGA mode
'some sample graphics
LINE (0, 0) - (639, 349), 1, B
LINE (10, 10) - (100, 40), 2, BF
CIRCLE (30, 30), 4, 4
PODPY 0, 1
CLS
FOR T = 1 TO 10
CALL FadeEGA (1, 1, T)
WEALL
```

The sequence is pseudo random and is specified by the data in FADEDATA.GW.

When a grid region is specified and a RandomPortion% is specified, it will transfer one-tenth of the screen image. Each grid region must be specified a total of ten times, each time with a different random portion specified to transfer the entire image to that region. The order for the RandomPortion at least once. random portions is not important, as long as every region receives each

this routine, the effect is much more pleasing to the eye and doesn't appear background of the two screen images is going to be similar. Any time when there is a large portion of the screen which will not be changed by is advisable to use this routine and this effect when the overall

Where:

speed. Including the

or modules which us All parameters for th Comments Row% and Colf r

The following exam

DEFINI A-Z

Syntax:

ASCITY = GetChard

the specified screen

Purpose:

The function GetCi

example for this routine. transfer a screen from the background, and should be utilized as an The BASIC subroutines NightFall and RandomFade use this routine to

See Also:

Assembly Routines

Fade2EGA, NightFall, RandomFade

ON THE MANAGED AND ASSESSED.

See Also:

character values into a interpret pixel pattern character value on the Note that this routin

Chank = GetCh, ra-

CALL OPTIM

ClearScreenArray

THE STATE OF THE S

GetCharacter%

Assembler function contained in GW.LIB

Purpose:

est when the overall into the charged by time I out be charged by the and doesn't appear

The function GetCharacter% returns the ASCII character value stored at the specified screen position.

■ Syntax:

ASCII% = GetCharacter%(BYVAL Row%, BYVAL Col%)

Where:

Row% and Col% represent the location on the screen to inquire about.

Assembly Routines

Comments:

e use this routine to Id be utilized as an All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The following example gets the character value at location 1,1.

DEFINT A-Z

\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's CREEN 12 'sets the monitor in VGA mode

CALL GPrintOVE (1, 1, "Hello", 15)

Char% = GetCharacter%(1, 1)

Note that this routine doesn't read video memory to determine the character value on the screen. To save the time that would be required to interpret pixel patterns, the Graphics Workshop Print routines put the character values into a screen array 80 characters wide by 25 lines high.

See Also:

ClearScreenArray

GetLastYCoor

GetLastXCoord%

function contained in GW.LIB Assembler

Purpose:

GetLastXCoord% returns the value of the last x coordinate drawn to

Syntax:

1% = GetLastYCoon

Purpose:

GetLastYCoord % r

Syntax:

X% = GetLastXCoord%

Comment:

of the drawing cursor's location after a Graphics Workshop primitive has been used. The following example demonstrates how to obtain the x coordinate value

been used. of the drawing curso The following exam Comment

OFFINT A-Z "\$INCLUDE: SCREEN 12

Assembly Routines

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' SCREEN 12 'required for BYVAL's 'sets the monitor in VGA mode

X% = GetLastXCoord%

CALL LineStep(103, 56, 15)

See Also:

■ See Also:

NX = GetLastYCoord

CALL LineStep[103,

GetLastXCoord"

GetLastYCoord%, SetLastCoord

ON SHIMMEN MANAGED

GetLastYCoord%

a ned in GWIIB

rdinate drawn to.

Assembler function contained in GW.LIB

Purpose:

GetLastYCoord% returns the value of the last y coordinate drawn to.

■ Syntax:

Y% = GetLastYCoord%

Comment:

le x coordinate value kshop primitive has

The following example demonstrates how to obtain the y coordinate value of the drawing cursor's location after a Graphics Workshop primitive has been used.

DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's SCREEN 12 'sets the monitor in VGA mode

ege

CALL LineStep(103, 56, 15)

Y% = GetLastYCoord%

See Also:

GetLastXCoord%, SetLastCoord

Assembly Routines

GMOVETVE

n purpose:

GetScreenMode%

Assembler function contained in GW.LIB

Purpose:

GetScreenMode% returns the BIOS video mode currently in use.

Syntax:

BIOSMode% = GetScreenMode%

Comments:

The following example checks to see if the BIOS video mode is greater than 13, meaning it is an EGA or VGA video mode.

Where:

FromCols, and Fro

column values range

Syntax:

CALL GHOVE WE (BY COIST, BYWA LI

on any video page. GMove I VE moves

Assembly Routines

SCREEN 12 DEFINT A-Z
'\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL's 'sets the monitor in VGA mode

IF BIOSMode% 13 THEN PRINT "EGA or VGA"

BIOSMode% = GetScreenMode%

ON SHIPLINGSON

destination pages can b

information and page 0 The default pages for

CALL GROVETVERIL, O, CALL SetGMPages(0, 0 page to the left half of

SCREEN 12 SCREEN 12

The following code fo

All parameters speed. Including the

or modules which pro

Comments:

the upper-left corner DestCol% and Dest an EGA monitor. These coordinates moved. Cols% and

GMove1VE

Elled in CMIIB

The Parties

Assembler subroutine contained in GW.LIB

Purpose:

GMove IVE moves any rectangular block on any video page to any location on any video page.

Syntax:

CALL GMovelVE (BYVAL FromCol%, BYVAL FromLine%, BYVAL Cols%, BYVAL DestLons%)

Where.

deo mode is greater

FromCol% and FromLine% specify the upper corner of the region to be moved. Cols% and Lines% specify the size of the region to be moved. These coordinates follow a mixed coordinate window system where column values range from 1 to 80 and line values range from 0 to 349 on an EGA monitor.

DestCol% and DestLine% specify a mixed coordinate value which tells the upper-left corner where the image will be placed.

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The following code fragment copies an image from the right half of the page to the left half of the page.

DEFINT A-Z '\$INCLUDE: 'GMDECL.BAS'' 'required for BYVAL's SCREEN 12 'sets the monitor in VGA mode

CALL SetGWPages(0, 0) 'make destination the same as the source

CALL GMove1VE(41, 0, 40, 480, 1, 0)

The default pages for this routine and others are page 1 for the source information and page 0 for the destination of the image. The source and destination pages can be re-directed using the SetGWPages routine.

Assembly Routines

GMove2VE

Purpose:

GMove2VE will sa

a video memory lo

When using the GMoveIVE routine to move an image across the screen, you must take into account that the image is being overwritten while it is being moved. For these uses, look at the use of the GMove2VE routine in DEMOMOVE.BAS and the GScrollVE routine in DEMOSCRL.BAS to move images flawlessly.

See Also:

Assembly Routines

Where:

These coordinates I moved. Cols% and FromCol% and Fro

a VGA monitor. column values rang

or VGA graphics r.

the information. The The variable L

The variable Direction

Syntax:

DestSegment% = 8H CALL GHOVECAE [BY Lines%, BYVAL D

less costly to the pr restore routines, an saves and restores memory to store the

GMove2VE, GMove3VE, TransferEGA, SetGWPages

■ 3 - 78

CRESCENT SOFTWARE, INC.

ON SHAMMEN MARKED

by 100 lines region of

The following example or modules which use speed. Including the f All parameters for this Comments: restored. Azero save

GMove2VE

IN DENOCAL BAS STATES WATER The Sales Sales

subroutine contained in GW.LIB Assembler

GMove2VE will save and restore any rectangular region of the screen to a video memory location which you specify. This routine uses screen memory to store the image. This approach has two advantages: Graphics saves and restores require one-fourth the instructions of other save and restore routines, and graphics memory is often not used and is therefore less costly to the programmer than using general memory.

Syntax:

DestSegment% = &HA800 CALL GNove2VE (BYVAL FromCol%, BYVAL FromLine%, BYVAL Cols%, BYVAL_ Lines%, BYVAL DestSegment%, BYVAL Direction%)

moved. Cols% and Lines% specify the size of the region to be moved. These coordinates follow a mixed coordinate window system where FromCol% and FromLine% specify the upper corner of the region to be column values range from 1 to 80 and line values range from 0 to 479 on a VGA monitor.

The variable DestSegment% provides the routine with a location to send the information. This segment value should be within the range of EGA or VGA graphics memory available. The variable Direction% decides whether the image will be saved or restored. A zero saves the image. Any other value will restore the image.

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The following example saves and restores the upper-left corner 10 column by 100 lines region of the screen.

Assembly Routines

```
'restore the image CALL GMove2VE (1, 0, 10, 100, &HAA00, -1)
                                                                                      CALL GMove2VE (1, 0, 10, 100, &HAA00, 0)
                                                                                                                                 LINE (0, 0) - (79, 99), 1, B
                                                                                                                                                                                SCREEN 12
                                                                                                                                                                                                 DEFINT A-Z
'$INCLUDE: 'GWDECL.BAS'
                                                                                                            'save the image
                                                                                                                                                                       'sets the monitor in VGA mode
                                                                                                                                                                                          'required for BYVAL's
```

SetGWPages. source for the image. The default page can be re-directed using the routine The default page for this routine is page 0. Depending upon the value in the parameter Direction%, the default page refers to the destination or the

Where:

an EGA monitor. These coordinates moved. Cols% and FromCol% and Fro

column values range

the upper-left come

DestCol% and Dest

Syntax:

CALL GHOVE 3VE (BY Linesk, BYVAL

purpose:

GMove3VE moves

effect by skipping a on any video page GMOVE3VE

Assembly Routines

&HA800. On the EGA display there is 128K free for the storage of The beginning of the EGA's high-resolution second screen starts at

segment would be &HAA00 as shown in the above code example. The VGA high-resolution mode doesn't have a second screen per se, but we still can use this routine. For the VGA the first unused graphics VGA has only 96K of available memory for the storage of images.

by an image saved with this routine: The following formula will help to calculate the amount of memory used

```
MemUsed% = Cols% * Lines% * 4
```

stored, use To use this to determine the next segment where graphics images can be

```
NextSegment% = ThisSegment% + MemUsed% \ 64 + 1
```

being stored. where ThisSegment% is the segment where the current graphics image is

The SkipLines parame or modules which use

speed. Including the t All parameters for this Comments: within the regn SkipLines % to

This routine is a vital part of saving graphics images for use in the graphics PullDownG menus and in the VertMenuG routine.

See Also:

GMove1VE, GMove3VE, GMove4VE, SetGWPages

ORIGENT WHITTER

Off Breams Inc

Sciplines" =

Tight half of the screen

Z-Y JAHARA

THE TOTAL STREET

The following example

display the opening titl graphics image. The o BASIC routine Splick bring in half of the lin every other line. Also

GMove3VF

subroutine contained in GW.LIB Assembler

Purpose:

GMove3VE moves any rectangular block on any video page to any location on any video page. This routine has the ability to create an interlacing effect by skipping a specified number of lines between lines transferred

CALL GHOVE3VE (BYVAL FromCol%, BYVAL Fromtine%, BYVAL Cols%, BYVAL Lines%, BYVAL DestCol%, BYV

Syntax:

ing upon the value in the destination or the ted using the routing

nd screen starts at for the storage of

Screen per se, but st unused graphics ode example. The

From Col% and From Line% specify the upper corner of the region to be moved. Cols% and Lines% specify the size of the region to be moved. These coordinates follow a mixed coordinate window system where column values range from 1 to 80 and line values range from 0 to 349 on an EGA monitor.

DestCol% and DestLine% specify a mixed coordinate value which tells the upper-left corner where the image will be placed. SkipLines% tells the routine how many lines to skip for every line copied within the region.

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The SkipLines parameter is usually set to 1 which tells the routine to skip every other line. Also, it is common for this routine to be called once to BASIC routine SplitHorizontal makes use of this routine to bring in a The demo DEMOGW.BAS makes use of this routine to bring in half of the lines and once more to bring in the other half. graphics image. The demo DI display the opening title page. The following example shows how to transfer an interlaced image of the right half of the screen to the left half of the screen.

'required for BYVAL's 'sets the monitor in VGA mode *INCLUDE: 'GWDECL.BAS' DEFINT A-Z

SkipLines% = 1 CALL GMove3VE (41, 0, 40, 480, 1, 0, SkipLines%)

Assembly Routines

of of memory used ics images can be e of images.

se in the graphics

CRESCENT SOFTWARE, INC.

GMOVE4VE

Purpose:

GMove4VE will sa

an array you specif

0 This routine works exactly like GMove1VE if you specify SkipLines % =

The default pages for this routine are page 1 for the source information and page 0 for the destination of the image. The source and destination pages can be re-directed using the routine SetGWPages.

See Also:

Assembly Routines

GMove1VE, GMove2VE, SetGWPages, SplitHorizontal, TransferEGA

Where:

Syntax:

CALL GNOVEAUE (B)

Linesk, BYVAL

CRESCENT SOFTWARE, INC.

THEORY OF THE DO Not styl styl bre death

Needed 8. If the value

Once the amount of I

NetaryNeededt = Colly Tomula:

array. To calculate th

The memory location or modules which use speed. Including the f All parameters for this

Comments: the image. restored. A value of

The variable Dire

follows:

REDIN Array

the information. Thi The variable DesiSe a VGA monitor. column values range These coordinates moved. Cols % and FromCol% and Fr

of a pre-dimension

And broggams with this you must run QB.EXE o

GMove4VE

ourse and destination

Assembler subroutine contained in GW.LIB

Purpose:

GMove4VE will save and restore any rectangular region of the screen to an array you specify.

■ Syntax:

onal Transferbox

CALL GMove4VE (BYVAL FromCol%, BYVAL FromLine%, BYVAL Cols%, BYVAL Lines%, BYVAL DestSegment%, BYVAL Direction%)

■ Where

moved. Cols% and Lines% specify the size of the region to be moved. These coordinates follow a mixed coordinate window system where FromCol% and FromLine% specify the upper corner of the region to be column values range from 1 to 80 and line values range from 0 to 479 on a VGA monitor.

Assembly Routines

The variable DestSegment% provides the routine with a location to send the information. This segment value is determined by finding the segment of a pre-dimensioned array. The segment of an array can be found as follows:

REDIM Array%(0 to 5000)
DestSegment% = VARSEG(Array%(0))

A value of zero saves the image. Any other value will restore The variable Direction% decides whether the image will be saved or restored.

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation. The memory location must be declared prior to saving the image into the To calculate the amount of memory required use the following formula:

MemoryNeeded% = ColumnsUsed% * LinesUsed% * 4 + 4

Once the amount of memory required has been calculated, you will dimension an integer array with half of the elements contained in Memory-Needed %. If the value in MemoryNeeded % is greater than 65536, then you must run QB.EXE or QBX.EXE with the /Ah parameter, and compile your programs with this parameter as well. In addition, you will need to create and pass this routine a long integer array where each element will

provide you with 4 bytes of memory space. To make an array covering 128K of memory, dimension it as follows:

REDIM LongArray&(0 to 32767)

100 lines high in the upper-left corner of the screen. The following example saves and restores a region 10 columns wide by

purpose:

GScroll VE scrolls

```
'restore the image CALL GMove4VE (1, 0, 10, 100, VARSEG(A%(0)), -1)
                                                                 CLS
                                                                                                  'save the image

MemNecded% = 10 * 100 * 4 + 4

DIM AX(MemNecded% \ 2) 'each integer counts for 2 bytes

CALL GMove4VE (1, 0, 10, 100, VARSEG(A%(0)), 0)

WHILE INKEY$ = "": WEND
                                                                                                                                                                                                                                   LINE (0, 0) - (79, 99), 1, B
                                                                                                                                                                                                                                                                                          '$INCLUDE: 'GWDECL.BAS' SCREEN 12
                                                                                                                                                                                                                                                                                                                                          DEFINT A-Z
                                                                                                                                                                                                                                                                                'sets the monitor in VGA mode
                                                                                                                                                                                                                                                                                                            'required for BYVAL's
```

Where:

CALL OSCIOTIVE

Assembly Routines

that if BASIC has moved your array, that you will be passing the proper address of your array to the GMove4VE routine. By calling VARSEG at the time when GMove4VE is called, you will ensure

ColDelta% and Lin

values range from I coordinates follow Cols% and Lines Coll % and Linel 9

The default page for this routine is page 0. Depending upon the value in the parameter Direction%, the default page refers to the destination or the source for the image. The default page can be re-directed using the routine SetGWPages.

cannot be displayed with the GMove4VE routine. statement. However, images captured with the BASIC GET statement routine can be placed anywhere on the screen using the BASIC PUT A side-effect of this routine is that images saved with the GMove4VE

See Also:

GMove1VE, GMove2VE, GMove3VE, SetGWPages

screens boundaries. You can use the elemen

CIRCLE (120, 120), 4, PAINT (120, 120), 4,

SCREEN TS. STOREGY F

DEF IN AN

Off Georgias

The following code fr.

combinations of these Not only can the wind

or modules which re speed. Includin All parameter Comments: negative direction to scrolled. A negativ

GScrollVE

15 columns wide by

Assembler Subroutine contained in GW.LIB

I Purpose:

GScrollVE scrolls a rectangular region on the screen.

■ Syntax:

CALL GScrollVE (BYVAL Coll%, BYVAL Linel%, BYVAL Cols%, BYVAL Lines%, BYVAL ColDelta%, BYVAL LineDelta%)

■ Where:

bytes

Coll% and Lines% specify the upper corner of the region to be moved. Cols% and Lines% specify the size of the region to be moved. These coordinates follow a mixed coordinate window system where column values range from 1 to 80 and line values range from 0 to 479 on a VGA monitor.

ColDelta% and LineDelta% specify the distance the region should be A negative number can be used to specify movement in the negative direction for that axis. scrolled.

led, you will ensure passing the proper

Comments:

e destination or the dusing the routine

upon the value in

h the GMove4VE

GET statement

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

Not only can the window be scrolled left, right, up and down, but diagonal combinations of these are also possible. The following code fragment scrolls a circle in a diagonal direction on the

DEFINI A-Z

'\$INCLUDE: 'GWDECL.BAS' 'required for BYVAL'S

SCREEN 12 'sets the monitor in VGA mode

CIRCLE (120, 120), 40, 15

PAINT (120, 120), 4, 15

FOR T = 1 TO 10

CALL GSGro1lVE(1, 0, 22, 169, 1, 8)

You can use the elements 43 and 44 of the GPDat%() array to check the screens boundaries.

Assembly Routines

Mulbylonitor

MakeAddressVE%

subroutine contained in GW.LIB **Assembler**

one bit set. This bit represents the pixel location within memory. MakeAddressVE% takes an (X, Y) coordinate and returns an EGA or VGA video memory address and a bit pattern. The bit pattern has only

Syntax:

Address% = MakeAddressVE%(BYVAL XPos%, BYVAL YPos%, BitPattern%)

Where:

Each bit in the retur

which is possible.

Syntax:

雅 = KultHonitor

This routine will te computer. It is pos MultMonitor% des

Assembly Routines

XPos% and YPos% specify the (X, Y) coordinate

one bit set which corresponds to the pixel position in memory. BitPattern% is a byte value returned from the assembly routine which has

Comment:

operation. all programs or modules which use this routine will insure proper Two parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of

location 100, 200 on the screen. The following example gets the information corresponding to pixel

```
'$INCLUDE: 'GWDECL.BAS' SCREEN 12
                                     DEFINT A-Z
required for BYVAL's sets the monitor in VGA mode
```

Address% = MakeAddressVE% (100, 200)

DrawByteVE routine. There are more optimal ways to set a pixel, but addressing memory directly is faster than by using X, Y coordinates. Having a memory location and a bit pattern, the pixel can be set using the

See Also:

DrawByteVE

ON STRUCTURAL NO.

course and the values p

Servideo combination

we suggested in Char

the best monitor availab The file GETVIDEO.E To check if a VGA mo

F (M AND 22) 00

Comments:

monitor connected will For example, asystem

MultMonitor%

aired in CALIB

function contained in GW.LIB Assembler

Purpose:

returns an EGA or bit pattern has only

the memory.

MultMonitor% determines all of the monitors attached to a particular computer. It is possible to have a computer with two or more monitors. This routine will tell you which types of monitors are active.

Syntax:

M% = MultMonitor%

V routine which has

memory.

ue to provide the it the beginning of 'Ill insure proper

Each bit in the return value represents an adapter and monitor combination which is possible. A value of 0 means no graphics monitor is attached.

Assembly Routines

MATERA MITERA	Horming	ricicules adaptor is attached	CGA capable adaptor attached	mono EGA adaptor is attached	color EGA adaptor is attached	mono VGA adaptor is attached	color VGA adaptor is attached	mono MCGA adaptor is attached	color MCGA adaptor is attached	EGA adaptor emulating CGA	IBM 8514/A adaptor is attached
VALITE	-		7	4	∞	16	32	64	128	256	512
BIT	0	,	-	2	3	4	S	9	7	∞	6

ponding to pixel

For example, a system which has both a VGA color monitor and a Hercules monitor connected will return a value of 33 (32 for VGA + 1 for Hercules). DIM 6314/A adaptor is attached

> n be set using the set a pixel, but

coordinates.

Comments:

To check if a VGA monitor exists, use the following line of code:

IF (M% AND 32) <> 0 THEN PRINT "Can use VGA"

The file GETVIDEO.BAS tests each bit in the proper order to determine the best monitor available. GETVIDEO.BAS is part of the standard code SetVideo combination will help to avoid any misunderstanding of this Also, using the GETVIDEO.BAS routine and the values it returns. we suggested in Chapter 1.

Assembly Routines

PaintBits

Assembler subroutine contained in GW.LIB

Purpose:

parameter allows you to mask out bits which you don't want re-colored a new color. For every occurrence of the old color in that byte of memory, PaintBits replaces it with the new color specified. A "bit pattern" PaintBits accepts a video memory location, a bit pattern, an old color and

PaintBits' and Paint to screen you do not we which you do not we

SCREEN IC SCREEN

Syntax:

CALL PaintBits(BYVAL Coll%, BYVAL Line1%, BYVAL BitPattern%, BYVAL OldColor%, BYVAL NewColor%)

of a specified color

PaintByte

Where

Assembly Routines

range from 1 to 80 and line values range from 0 to 479 on a VGA monitor. coordinates follow the mixed coordinate system where column values Coll% and Line1% specify the location of a byte in memory.

the ability to be changed BitPattern% is a byte where every bit which is on (a binary value 1) has

OldColor% is the color which exists on the screen that we want to change.

NewColor% is the color to replace the former color.

Comments:

or modules which use this routine will insure proper operation. speed. Including the file GWDECL.BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

which does not accept a bit pattern. If you set BitPattern% to 255, this function works exactly like PaintByte

region: red, white, and blue. If you change the white to a blue, but then originally blue to white. want to change it back to white, you will end up changing that which was undesirable results. For example, imagine there are three colors in a Using this function to change the color back to the original can produce

bits on the screen, to blue if and only if they are black The example below changes every other bit among the first eight horizontal

Eigh Oline

Assembler

ten, an id color and

d. A "bit paten" n't want re-colored.

NewColor% = 1 'blue 'black '01dColor% = 0 'black 'black BitPattern% = 170 'every other bit set CALL PaintBits (1, 0, BitPattern%, 01dColor%, NewColor%) 'required for BYVAL's 'sets the monitor in VGA mode DEFINT A-Z
'\$INCLUDE: 'GWDECL.BAS'
SCREEN 12

screen you want to re-color has colors which exist outside the region, which you do not want to re-color. If it is acceptable for all occurrences of a specified color to be changed, then change its palette instead. PaintBits' and PaintByte's real power shows up when the region of the

See Also:

PaintByte

in memory. The Pere column values on a VGA monitor. binary value 1) has

Patterny,

we want to change.

tly like PaintByte

vide the maximum

ng of all programs

eration.

inal can produce three colors in a

a blue, but then g that which was t eight horizontal

Assembly Routines

CRESCENT SOFTWARE, INC.

T SOFTWARE, DIC

3 - 89

PaintByte

subroutine contained in GW.LIB Assembler

See Also: of a specified color screen fou want to PaintBits' and Paint

which you do not w

PaintBits

Purpose:

replaces it with the new color specified. color. PaintByte accepts a byte location on the screen, an old color and a new For every occurrence of the old color on the screen, PaintByte

Syntax:

CALL PaintByte(BYVAL Coll%, BYVAL Line1%, BYVAL OldColor%, BYVAL NewColor%)

Where:

Assembly Routines

range from 1 to 80 and line values range from 0 to 479 on a VGA monitor. coordinates follow the mixed coordinate system where column values Coll% and Linel% specify the location of a byte in memory. The

OldColor% is the color which exists on the screen that we want to change

NewColor% is the color to replace the former color.

Comments:

or modules which use this routine will insure proper operation. speed. Including the file GWDECL.BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

originally blue to white. want to change it back to white, you will end up changing that which was undesirable results. For example, imagine there are three colors in a region: red, white, and blue. If you change the white to a blue, but then Using this function to change the color back to the original can produce

to red if they are black. The example below changes the first eight horizontal bits on the screen,

DEFINT A-Z *\$INCLUDE: 'GWDECL.BAS'

SCREEN 12 'required for BYVAL's 'sets the monitor in '

in VGA mode

NewColor% = 4 OldColor% = 0

ON STREET, STREET, WASTER

Raned in GWLIB

noid coll and a new the screen PainBue

PaintBits' and PaintByte's real power shows up when the region of the screen you want to re-color has colors which exist outside the region, which you do not want to re-color. If it is acceptable for all occurrences of a specified color to be changed, then change its palette instead.

See Also:

PaintBits

dSolong.

Assembly Routines

e in memory. The here column values On a VGA monitor. t we want to change.

ovide the maximum ing of all programs three colors in a to a blue, but then ng that which was

its on the screen.

ginal can produce

peration.

CRESCENT SOFTWARE, INC.

3 - 91

Assembly Routines

ScrnDump2

subroutine contained in GW.LIB Assembler

Purpose:

of printers can be printed in either landscape or portrait mode. mode, to either an Epson or compatible dot matrix printer, or to a Hewlett ScrnDump2 will print a snapshot of a graphics screen, regardless of the Packard LaserJet Series II compatible printer. The images for both types

CALL ScrnDump2 (DPI\$, LptNumber%, Translate%, XMult%, YMult%, LandOrPort%)

Where:

Assembly Routines

on an Epson FX series or compatible printer. The values are "075", "100", "150" and "300" DPI\$ indicates the Dots Per Inch resolution when sending to a Hewlett-Packard LaserJet or compatible printer, or is a null string ("") if printing The valid strings for the

LPTNumber% set to -1. port to use. LptNumber% is either 1, 2, or 3, to tell ScrnDump2 which parallel printer If a printer error occurs, ScrnDump2 will return

resolution when sent

This will make the i IF LptNoX = -1 THE CALL Scrinburg2 ("1 LotNoX = 1

ratio on the HP Lase

not always possible. buffer is only 720 b YMult% variables. There are some res

Store Laser Jet inform

the demonstration p

There are many exa

OKL Scroburge (" I = KONIAT

to equivalent tile patterns, or it is set to 0 to print all colors as solid black. Translate% is set to any non-zero value to translate all colors on screen

the values in these variables are left alone by the routine. pre-set to give the proper aspect. When printing to an EPSON style printer to the LaserJet printers. All aspects for the EPSON style printers are XMult% and YMult% are aspect multipliers for images which are sent

print the image in Landscape mode. A zero value will print in Portrait will be displayed in Landscape or Portrait mode. Any non-zero value will LandOrPort% is a true or false variable which determines if the image

Comments:

corner of the image at the printer's current cursor position. When printing on a laser printer, ScrnDump2 positions the upper-left

determine the number of screen bytes being used and their organization. ScrnDump2 will automatically recognize the current video mode, and

at 150 dots per inch in Landscape mode is given below. An example of printing a VGA high-resolution screen to an Laser Printer

CRESCENT SOFTWARE, INC.

Laned in Civilia

teen teganless of the printer, or to a Hewlett images for both types

rtrait mode.

E. Mate.

'required for BYVAL's 'sets the monitor in VGA mode DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' LPTNo% = 1 SCREEN 12

LandOrPort% = -1 CALL ScrnDump2 ("150", LPTNO%, -1, 1, 1, LandOrPort%)

There are many examples of using the ScrnDump2 routine contained in There are some restrictions on the valid values for the XMult% and YMult% variables. A buffer inside the ScrnDump2 routine is used to store LaserJet information as it is being flipped into landscape mode. This buffer is only 720 bytes long. Trying to double the size of an image is not always possible. To give a 640 by 200 pixel screen to proper aspect ratio on the HP LaserJet, follow the example below: the demonstration program SCRNDUMP.BAS.

> ending to a Hewlett. tring ("") if printing valid strings for the

This will make the image appear like a screen with a 640 by 400 pixel resolution when sent to the printer.

hich parallel printer ump2 will return

ges which are sent

style printers are

200N style printer

on-zero value will Il print in Portrait

nines if the image

ns the upper-left

video mode, and

eir organization.

il colors on screen

Assembly Routines

Assembly Routines

The maximum XMult% and YMult% values are shown in the table below:

The Graphics Workshop

SetDestPage

purpose:

SelDestPage sets the drawing primitives.

Assembly Routines

11	10	9	00	7	ယ	2		SCREEN mode
1	_	_	-	-	_	-	2	XMult%
1	-	_	2	_	-	2	2	YMult%

Where:

DestPage% is the di

is page 0. The EGA Comments:

The following orde "

SCUEEN 8 ANDLINE: , 80.5

speed. Including the or modules which us The parameter for th Syntax:

CALL SetDestPage

ON 38 WASHING HASKED

■ See Also:

This routine does not a address of an EGA pa The VGA does not have

OALL LineME (NO., 10 CALL SetDestRoje (

SetGWPages, SetSoute

To the Paris of th

SetDestPage

subroutine contained in GW.LIB Assembler

Purpose:

SetDestPage sets the destination video page for image moves and for all drawing primitives.

Syntax:

CALL SetDestPage (BYVAL DestPage%)

Where:

The first video page DestPage% is the destination video page to be set. The first vid is page 0. The EGA has a video page 1, but the VGA does not.

Assembly Routines

Comments:

The parameter for this routine is passed by value to provide the maximum speed. Including the file GWDECL. BAS at the beginning of all programs or modules which use this routine will insure proper operation.

The following code will draw a line on the hidden video page.

'required for BYVAL's 'sets the monitor in EGA mode DEFINT A-Z '\$INCLUDE: 'GWDECL.BAS' SCREEN 9

CALL SetDestPage (1)

CALL LineVE (100, 100, 200, 200, 15)

The VGA does not have a video page 1, but this routine will mimick the address of an EGA page 1 when used on a VGA.

This routine does not affect any of BASIC's graphics statements.

See Also:

SetGWPages, SetSourcePage

SetGWPages

subroutine contained in GW.LIB Assembler

Purpose:

routines. SetGWPages sets the video pages used by the family of graphics move

Syntax:

CALL SetGWPages (BYVAL SourcePage%, BYVAL DestPage%)

Where:

LastY% is the You LastX% is the X co Syntax:

CALL SetLastCoo

SeiLastCoord sets ti

does it without mod

Assembly Routines

SourcePage% is the page number for the source of these routines. Valid Page numbers for the EGA are 0 and 1.

Page numbers for the EGA are 0 and 1. DestPage% is the page number for the destination of these routines. Valid

Comments:

all programs or modules which use this routine will insure proper maximum speed. Both parameters for this routine are passed by value to provide the Including the file GWDECL.BAS at the beginning of

> or modules which us speed. Including the

draw a line 10 p

DEFINI A-Z

The following exam

All parameters for th

Comments:

both point to the first video page: The following code fragment makes the Source Page and Destination Page

```
SCREEN 12
                          '$INCLUDE: 'GWDECL.BAS'
'sets the monitor in VGA mode
                      'required for BYVAL's
```

DEFINT A-Z

CALL SetGWPages (0, 0)

This routine does not modify the visual and active pages.

an error, but will address memory as if the EGA video mode was active. the only available page is page 0. Using a page value of 1, will not cause The VGA does not have enough memory to have 2 video pages. Therefore, You can use this knowledge to your advantage with some insight.

See Also: GetLastXCoord%, Get

will be very useful in i operators. If the Lines screen, is apparent wh The benefit that this r

CALL LineToStep (9,

It is allowed to make the source and destination pages the same.

video pages individually. The routines SetDestPage and SetSourcePage will allow you to set the

SetDestPage, SetSourcePage

ON STRUCTURE MANAGED

SetLastCoord

tained in GWIIB

Assember

y of graphics more

subroutine contained in GW.LIB Assembler

Purpose:

SetLastCoord sets the coordinate which is the last coordinate drawn to. It does it without modifying any pixel values on screen.

Syntax:

CALL SetLastCoord (BYVAL LastX%, BYVAL LastY%)

Where:

LastX% is the X coordinate value to be set.

LastY% is the Y coordinate value to be set.

these routing. Valid

alue to provide the at the beginning of will insure proper

Assembly Routines

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation. The following example sets the coordinate value to (103, 53) and then draw a line 10 pixels long to the right.

\$INCLUDE: 'GWDECL.BAS'

d Destination Page

SCREEN 12

'required for BYVAL's 'sets the monitor in VGA mode

CALL SetLastCoord (103, 53)

CALL LineToStep (9, 0, 15)

screen, is apparent when the line step routines are used with their logical operators. If the LineStep or LineToStep routines are required this routine The benefit that this routine does not affect any of the pixel values on will be very useful in implementing the task.

See Also:

pages. Therefore, of I, will not cause

mode was active.

ne insight. the same. W you to set the

GetLastXCoord%, GetLastYCoord%

NT SOFTWIRE DIE

SlideDown

Purpose:

onto the screen at a SlideDown takes a l

SetSourcePage

subroutine contained in GW.LIB Assembler

Purpose:

SetSourcePage sets the source video page for graphics moves.

Syntax:

CALL SetSourcePage (BYVAL SourcePage%)

page 0. The EGA has a video page 1, but the VGA does not. SourcePage% is the source video page to be set. The first video page is

Where:

LRCol% and LRL ULCol% and ULLir

follow the mixed coo region. Together the

80, and line values to

Syntax:

CALL SlideDown(BY)
BYVAL LRLineX.

Comments:

or modules which use this routine will insure proper operation. The parameter for this routine is passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs

example. To set the source video page on an EGA screen to page 1, follow the below

the upper-left corner DestCol% and DestI

All parameters

Comments:

'\$INCLUDE: 'GWDECL.BAS' DEFINT A-Z

'required for BYVAL's 'sets the monitor in EGA mode

CALL SetSourcePage(1)

address of an EGA page 1 when used on a VGA The VGA does not have a video page 1, but this routine will mimick the

This routine does not affect any of BASIC's graphics statements.

See Also;

For an example of how program.

or modules which pr

SlideUp, SlideLeft, Sli

3 See Also:

SetDestPage, SetGWPages

SlideDown

Sahed in GWIB

subroutine contained in GW.LIB Assembler

Purpose:

SlideDown takes a region of the second video page screen and slides it onto the screen at a specified location in a downward direction.

Syntax:

he first video page is

does not.

CALL SlideDown(BYVAL ULCol%, BYVAL ULLine%, BYVAL LRCol%, BYVAL LRLine%, BYVAL DestCol%, BYVAL DestLine%)

Where:

rovide the maximum ning of all programs

1, follow the below

Operation,

ULCol% and ULLine% specify the upper-left corner of the source region. LRCol% and LRLine% specify the lower-right corner of the source region. Together they specify the region to be moved. These coordinates follow the mixed coordinate system where column values range from 1 to 80, and line values range from 0 to 349 on an EGA monitor.

DestCol% and DestLine% specify a mixed coordinate value which tells the upper-left corner of the image's final resting place.

Comments:

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

For an example of how to use this routine, see the DEMOFX.BAS example program.

See Also:

statements.

ne will mimick the

SlideUp, SlideLeft, SlideRight

Assembly Routines

SlideLeft

subroutine contained in GW.LIB Assembler

purpose:

SlideRight takes a f

side of the screen. screen at a specifie

Purpose:

of the screen. screen at a specified location in a horizontal direction towards the left side SlideLeft takes a region of the second video page and slides it onto the

Syntax:

CALL SlideRight(8

Where:

ULCol% and ULLin

LRCol% and LRL

follow the mixed coo region. Together the

Syntax:

CALL SlideLeft(BYVAL ULCol%, BYVAL ULLine%, BYVAL LRCol%, BYVAL LRLine%, BYVAL DestCol%, BYVAL DestLine%)

Where:

Assembly Routines

region. Together they specify the region to be moved. These coordinates 80, and line values range from 0 to 349 on an EGA monitor. follow the mixed coordinate system where column values range from 1 to LRCol% and LRLine% specify the lower-right corner of the source ULCol% and ULLine% specify the upper-left corner of the source region.

the upper-left corner of the image's final resting place. DestCol% and DestLine% specify a mixed coordinate value which tells

Comments:

speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation. All parameters for this routine are passed by value to provide the maximum

program. For an example of how to use this routine, see the DEMOFX.BAS example

See Also:

SlideDown, SlideUp, SlideRight

See Also:

For an example of how program.

speed. Including the

or modules which

All parameters in

Comments:

the upper-left corner

DestCol% and Dest 80, and line values of

SlideDown, Slidelly, S

OSCIENTANTAME NO.

SlideRight

La Tied in CIVILIB

Assembler subroutine contained in GW.LIB

Purpose:

and slice it one the town the

SlideRight takes a region of the second video page and slides it onto the screen at a specified location in a horizontal direction towards the right side of the screen.

■ Syntax:

CALL STideRight(BYVAL ULCol%, BYVAL ULLine%, BYVAL LRCol%, BYVAL LRLine%, BYVAL DestCol%, BYVAL DestLine%)

■ Where:

of the source region.

J. These coordinates these range from 1 to

ate value which tells

Nonitor.

ULCol% and ULLine% specify the upper-left corner of the source region. LRCol% and LRLine% specify the lower-right corner of the source region. Together they specify the region to be moved. These coordinates follow the mixed coordinate system where column values range from 1 to 80, and line values range from 0 to 349 on an EGA monitor.

DestCol% and DestLine% specify a mixed coordinate value which tells the upper-left corner of the image's final resting place.

Comments:

ovide the maximum ing of all programs

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

For an example of how to use this routine, see the DEMOFX. BAS example program.

See Also:

SlideDown, SlideUp, SlideLeft

Assembly Routines

SlideUp

Assembly Routines

subroutine contained in GW.LIB Assembler

purpose:

SplitHorizontal take

SolitHorizonta

Purpose:

SlideUp takes a region of the second video page and slides it onto the screen at a specified location in an upward direction.

Syntax:

CALL SlideUp (BYVAL ULCol%, BYVAL ULLine%, BYVAL LRCol%, BYVAL LRLine%, BYVAL DestLone%)

Syntax:

CALL SplitHorizon

image which appear line of the image wi the screen at a spec

Where:

ULCol% and ULLir

region. Together the LRCol% and LRL

the upper-left cor

DestCol% and Dest 80, and line values r ollow the mixed coo

All parameters for the Comments:

Where:

Assembly Routines

follow the mixed coordinate system where column values range from 1 to 80, and line values range from 0 to 349 on an EGA monitor. region. Together they specify the region to be moved. These coordinates LRCol% and LRLine% specify the lower-right corner of the source ULCol% and ULLine% specify the upper-left corner of the source region.

the upper-left corner of the image's final resting place. DestCol% and DestLine% specify a mixed coordinate value which tells

Comments

or modules which use this routine will insure proper operation. All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL BAS at the beginning of all programs

program. For an example of how to use this routine, see the DEMOFX.BAS example

See Also:

SlideDown, SlideLeft, SlideRight

See Also:

For an example of how or modules which use speed. Including the f

SlideDown, Slidelly, S

SplitHorizontal

is ned in OVIUB

and slides it amo the

subroutine contained in GW.LIB Assembler

SplitHorizontal takes a region of the second video page and slides it onto the screen at a specified location in a horizontal direction. Every other ine of the image will come from the opposite direction. The result is an image which appears to weave itself together.

Syntax:

r of the source region. corner of the source d. These coordinates alues range from 1 to

CALL SplitHorizontal (BYVAL ULCol%, BYVAL ULLine%, BYVAL LRCol%, BYVAL DestCol%, BYVAL DestLine%)

Assembly Routines

LRCol% and LRLine% specify the lower-right corner of the source ULCol% and ULLine% specify the upper-left corner of the source region. region. Together they specify the region to be moved. These coordinates follow the mixed coordinate system where column values range from 1 to 80, and line values range from 0 to 349 on an EGA monitor.

ate value which tells

Monitor

DestCol% and DestLine% specify a mixed coordinate value which tells the upper-left corner of the image's final resting place.

Comments:

ovide the maximum ing of all programs

OFX. BAS example

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs or modules which use this routine will insure proper operation.

For an example of how to use this routine, see the DEMOFX. BAS example

See Also:

SlideDown, SlideUp, SlideRight, SlideLeft.

AT SOFTWARE DIC

TransferEGA

Assembler subroutine contained in GW.LIB

GMovel VE, GMov

The TransferEGA demonstrated in the

Purpose:

another video page. This routine is meant for use with EGA adapters. TransferEGA moves any block on any video page to the same location on

Syntax:

CALL TransferEGA (BYVAL FromCol%, BYVAL FromLine%, BYVAL _ Cols%, BYVAL Lines%)

■ Where:

Assembly Routines

an EGA monitor. column values range from 1 to 80 and line values range from 0 to 349 on FromCol% and FromLine% specify the upper corner of the region to be moved. Cols% and Lines% specify the size of the region to be moved. These coordinates follow a mixed coordinate window system where

Comments:

or modules which use this routine will insure proper operation. speed. Including the file GWDECL.BAS at the beginning of all programs All parameters for this routine are passed by value to provide the maximum

assumed to be the same location as on the source page. parameters for the destination location on the screen. The destination is This routine works like the GMovel VE routine, except that it doesn't have

destination pages can be re-directed using the SetGWPages routine. information and page 0 for the destination of the image. The source and The default pages for this routine and others are page 1 for the source

page to the first video page: The following example copies a full screen image from the second video

CHARGEL SHAMME INC.

faned Asembler GN/UB

The TransferEGA routine is used by the BASIC DoSegueX routines demonstrated in the example program QSEGUE.BAS.

■ See Also: GMove1VE, GMove2VE, GMove3VE, SetGWPages

the same location on with EGA address.

BYW

et of the region to be region to be moved ndow system where the from 0 to 349 on

rovide the maximum ning of all programs

that it doesn't have The destination is

peration

e. The source and

ages routine.

e 1 for the source

n the second video

Assembly Routines

CRESCENT SOFTWARE, INC.

NT SOFTWARE INC

3 - 105

Section 4: **Mouse Routines**

Application Using A Mouse With A Graphics Mode

is done with the H

any one time. The internal operation of a mouse works as follows: A Mouse works differently when operating in graphics mode than it does in text mode. The mouse covers, on average, 256 pixels on the screen at

ت When the mouse cursor is drawn on the screen by the mouse driver, the following occurs:

statements.

101

Assembly Routines

- The square region encompassing where the mouse cursor will be placed is saved by the mouse routines.
- <u>5</u> A mask is placed over this region which clears (makes black) all pixels that will be beneath the mouse cursor.
- square region. The actual mouse cursor is drawn on the screen filling the
- 2) When the mouse is removed from the screen, the following occurs:

are made to Hid

called DrawEye. Dr For an example of h screen processing. ShowCursor, because times as possible. will find that it is si the mouse cursor o Each time you call S register which cour communicate with must temporarily n For graphics mode active is simple; jui mouse is active.

following: You cres

DrawPerson will ind

he count maintained

- replaced by the image previously saved. The square region where the mouse is now located will be
- ω When the mouse is physically moved, the following happens:
- a) (under #2). The mouse is removed from the screen as described above
- <u>b</u> The mouse is drawn at the new location as described above (under #1).

effectively losing the information we wanted to draw to the screen. the image saved before the line was drawn will be placed on the screen, draw a line (or anything) over the mouse. If the mouse then is moved, especially since the mouse is moved often. What concerns programmer is the last of these three mouse functions, The problem occurs if we

mouse while it is drawing the boxes. To see the problem, load the demo DEMOMOUS.BAS and move the demo then shows you the proper method for drawing graphics while the and when you move the mouse, a previous color appears in its place. The Notice that the mouse disappears

JAN BAWAHO MADORD

mouse is active. Programming for character mode while the mouse is active is simple; just turn it on and monitor it. For graphics mode, every, repeat EVERY, time you draw anything, you must temporarily turn off the mouse and turn it on again afterwards. This is done with the HideCursor and ShowCursor routines. These routines communicate with the mouse driver. The mouse driver has a special Each time you call ShowCursor the driver decreases this value, and places the mouse cursor on the screen only when this value reaches zero. You will find that it is simpler and more beneficial to call HideCursor as few register which counts the number of times you have called HideCursor. times as possible, such as at the beginning of a long list of LINE statements. You should be sparing in your use of HideCursor and ShowCursor, because each time you do it, it will take time out of normal screen processing.

> es mode than it does wels on the screen at

le mouse cursor will

reen by the mouse

orks as fallows:

lears (makes black)

ULYOU.

e screen filling the

m, the following

ny located will be

s described above

s described above

owing happens:

lem occurs if we

mouse functions. se then is moved. ed on the screen, S and move the in its place. The aphics while the

the screen.

nouse disappears

Assembly Routines

For an example of how HideCursor and ShowCursor work, consider the following: You create a routine called DrawPerson which calls a routine the count maintained by the mouse driver is needed. If a total of two calls are made to HideCursor, it will takes two calls to ShowCursor to restore called DrawEye. DrawEye can be called directly to make the eye blink, so it has to have a call to HideCursor and ShowCursor inside it. DrawPerson will have already hidden the mouse cursor. This is where the mouse onto the screen.

CRESCENT SOFTWARE, INC.

3 - 107

ButtonPress

Assembly Routines

subroutine contained in GW.LIB Assembler

purpose:

mouse buttons are GetCursor reports GetCursor

Purpose:

pressed coordinates where the mouse cursor was located when the button was last pressed since the last time it was called. It also returns the (X, ButtonPress will report how many times a specified mouse button was

CALL ButtonPress(Button%, Status%, Count%, X%, Y%)

Where:

CALL GetCursor(A)

Status% is bit-code X% and Y% return

bit 0 being on to ind The button informat

so forth. The vario

CALL GetDurson(XX.

Assembly Routines

button 2, and 3 for button 3 (if the mouse has a third button). Button% is the button of interest, with a 1 indicating button 1, 2 meaning

Status% is the current button status, and has the same meaning as the information returned by the GetCursor mouse routine.

Press was last called. Count% tells how many times the button has been pressed since Button-

X% and Y% hold the mouse cursor position at the time the button was location. pressed. Use the GetCursor routine to determine the current mouse cursor

Comments:

IF Status% AND 2 IF Status% AND

Comments:

a loop. report the current button status, it would have to be polled repeatedly in ButtonPress is the only reasonable way to determine when the mouse buttons are active and need attention. Though the GetCursor routine will

screen, both the X ar

example, if the mous that are available The X% and Y ...

moves one character b

code for both PullDownG and VertMenuG. A good example of implementing ButtonPress can be found in the source

tion of interpreting the X and Y values that are returned. The comments that accompany the GetCursor routine provide an explana-

to zero each time it is called. Note that unlike GetCursor, ButtonPress resets the button-press counter

See Also:

GetCursor

CHANGE MACKED AND MACKED

See Also: ButtonPress

many times the buttons Unlike the ButtonPress by 350, then the mouse

In graphics modes then same way but the Y re of a CGA high-resolut text screen is consider case the Y value sudde The same thing happe

GetCursor

taned in CIVIIB

ed mouse buton was returns the (X, Y) en the button was last

subroutine contained in GW.LIB Assembler

Purpose:

GetCursor reports the current location of the mouse cursor and which mouse buttons are currently depressed.

CALL GetCursor(X%, Y%, Status%)

Where:

button 1, 2 meaning

ome meaning as the

essed since Button-

X% and Y% return holding the current mouse cursor coordinates.

Status% is bit-coded to indicate which buttons are currently down.

Assembly Routines

bit 0 being on to indicate that button 1 is pressed, bit 1 for button 2, and so forth. The various bits may be easily isolated as shown below: The button information is represented by bits in the Status % variable, with

CALL GetCursor(XX, YX, StatusX)
IF StatusX AND 1 THEN PRINT "Button 1 is pressed"
IF StatusX AND 2 THEN PRINT "Button 2 is pressed"
IF StatusX AND 4 THEN PRINT "Button 3 is pressed"

rrent mouse cursor

me the button was

Jursor routine will

when the mouse lled repeatedly in

The X% and Y% values returned depend in part on the number of pixels example, if the mouse cursor is currently at the upper-left corner of the moves one character box to the right, the X Value will immediately become that are available on the screen. This is true even in text mode. screen, both the X and Y values will be returned as zero.

case the Y value suddenly jumps to eight. Thus, the resolution of a 25-line text screen is considered (to the mouse anyway) as being the same as that The same thing happens when the cursor is moved downward, in which of a CGA high-resolution display. The 43 line character mode works the same way but the Y resolution in the 50 line mode jumps in steps of 7. In graphics modes there are no jumps. If a screen has a resolution of 640 by 350, then the mouse could point to any one of the possible combinations.

Unlike the ButtonPress routine, GetCursor does not reset the count of how many times the buttons have been pressed.

See Also:

ton-press counter

ButtonPress

GrafCursor

subroutine contained in GW.LIB Assembler

Purpose:

GrafCursor greatly simplifies defining the shape of the mouse cursor for use in graphics mode.

Syntax:

CALL HideCurson

cursor before exper

One very important before drawing som prompt. Also, for common courtesy t Any program that i Comments: purpose:

HideCursor turns C

HideCursor

Syntax:

CALL GrafCursor(X%, Y%, Cursor\$)

Where:

X% and Y% define the cursor "hot spot"

Assembly Routines

DEMOMOUS.BAS. new cursor shape. Cursor\$ is either a conventional or fixed length string that contains the Examples of setting up this string are shown in

Comments:

how the cursor will appear. how GrafCursor can be used with a "pictorial" layout to quickly visualize The example in the DEMOMOUS.BAS demonstration program shows

location. Again, the example in DEMOMOUS.BAS shows this in context. at one time, only one point can be considered to be the actual cursor in graphics mode. Even though the mouse cursor will span several pixels The hot spot indicates which pixel the mouse is considered to be on when

cursor moves, the mo what you drew.

ucheath itself. If you

are drawing. In graph

mouse, it is necess

in graphics mo wu will need to times it was turned (BASIC LOCATE driver. Unlike the is how the current o

CHECKELL SHEWARE INC.

the cursor when it is fin it should turn off the mo routine which has alrea a routine to draw an ey

mouse. For example, a to be called from ano

due to the nature of gra cursor off before draw graphics on the screen HideCursor routine va screen. It is for this re applications turn the n This is why you see the

HideCursor

a ed in CWIIB

Assembler subroutine contained in GW.LIB

Purpose:

HideCursor turns off the mouse cursor.

■ Syntax:

CALL HideCursor

Comments:

ng that contains the tring are shown in

Also, for graphics programming, you must turn the mouse off Any program that is to be "mouse aware" will need to turn on the mouse common courtesy to turn it off again before returning them to the DOS Likewise, it is only cursor before expecting a user to access the mouse. before drawing something on the screen.

One very important point to be aware of regarding the HideCursor routine is how the current on and off status is maintained internally by the mouse Unlike the normal text cursor that is turned on or off with the BASIC LOCATE command, the mouse cursor keeps track of how many times it was turned on or off. Thus, if you call HideCursor twice in a row, you will need to call ShowCursor twice before it will be visible again.

> on program shows to quickly visualize

red to be on when pan several pixels the actual cursor ws this in context.

In graphics mode, when you want to draw something at the location of the mouse, it is necessary to turn off the mouse cursor temporarily while you are drawing. In graphics mode, the mouse has a copy of the screen image beneath itself. If you draw over the cursor with the cursor on, when the cursor moves, the mouse driver will re-draw the previous image, without what you drew. This is why you see the mouse flicker in large graphics applications. These applications turn the mouse off and on many times while drawing to the screen. It is for this reason that the above mentioned characteristic of the HideCursor routine can be useful. If you have multiple routines drawing graphics on the screen, it is necessary that each routine turn the mouse to be called from another routine which has previously turned off the due to the nature of graphics programming, a routine cannot always expect mouse. For example, a routine designed to draw an entire face might call a routine to draw an eye. If the eye routine were to be called separately, it should turn off the mouse cursor itself. If it is called from within another routine which has already turned off the cursor, then it should not turn on the cursor when it is finished. Instead the count maintained by the mouse cursor off before drawing and turn it back on before leaving.

Assembly Routines

InitMouse

Purpose:

to reset the mouse d laitMouse is used to

Where:

HaveMouse% rece

at the start of a progr

its travel range and st Because InitMouse to Comments: installed.

mouse driver of the mouse have installed before a now Understand that Ir.' Syntax:

CAL InitHouse(Ha

driver is merely decremented when the eye routine calls ShowCursor to turn the cursor back on.

See Also:

ShowCursor.

Assembly Routines

■ 3 - 112

CRESCENT SOFTWARE, INC.

TENENT NATURALE DC

InitMouse

Assembler subroutine contained in GW.LIB

Purpose:

InitMouse is used to determine if a mouse if present in the host PC, and to reset the mouse driver software to its default values.

■ Syntax:

CALL InitMouse(HaveMouse%)

■ Where:

HaveMouse% receives -1 if a mouse is present, or 0 if no mouse is installed.

Comments:

Because InitMouse resets the mouse driver values (the mouse cursor color, its travel range and sensitivity, etc.), it would probably be called only once at the start of a program. Understand that InitMouse doesn't actually detect the physical presence Rather, the mouse driver software must be installed before a mouse will be detected. Newer versions of Microsoft's mouse driver software actually detect if the mouse is physically attached to the machine, and will not load the driver unless the mouse is connected. of the mouse hardware.

Motion

Assembly Routines

subroutine contained in GW.LIB Assembler

Purpose:

to use those that ar

Workshop.

Mouse provides acc

Mouse

Purpose:

motion. Motion allows a program to establish the sensitivity of the mouse cursor

Syntax:

CALL Motion(Value%)

Where:

Assembly Routines

being the most sensitive. Value% is the desired sensitivity ranging between 1 and 32767, with 1

Where:

and DX% assign and

AV% is the number

Comments:

Syntax:

CALL Mouse(AXX, 8)

Comments:

with Graphics Workshop like this: values independently, you should use the generic Mouse routine provided yet another passed parameter. If you absolutely must be able to set these seems to be the most logical way to control a mouse, while eliminating vertical sensitivity separately, Motion uses the same value for both. This Even though the mouse driver software allows setting the horizontal and

> may be occasions wh Mouse provides acce

ones are provided as

CALL Mouse(15, 0, X%, Y%)

respectively. 15 represents the mouse service for sensitivity setting. Where X% and Y% represent the sensitivity for the X and Y coordinates

beyond 100 are hopelessly insensitive The stated upper range for the motion sensitivity is 32767; however, values

> can retrieve the m use of the mouse driver" needs to be call

the mouse's envi For instance, if

the mouse a "Mickey" You may be interested to know that Microsoft calls the unit of distance for

See Also:

Mouse

See Also: GetCountLocations, (

in context is shown in t

example is contained in retrieve the environme Location% and Get/10

session, this byte will the two environments u the count) and then ret the mouse environmer and ShowCursor is no

Mouse

Ga red in GWIB

Assembler subroutine contained in GW.LIB

Purpose:

A of the mouse cursor

Mouse provides access to all of the mouse services, and is the only way to use those that are not provided in a simplified form with Graphics Workshop.

Syntax:

CALL Mouse(AX%, BX%, CX%, DX%)

■ Where:

1 and 32767, with 1

AX% is the number for the mouse service of interest, while BX%, CX%, and DX% assign and return the processor's registers.

Comments:

S the horizonial and velue for both. This s, while eliminating be able to set these SE routing provided

and Y coordinates

itivity setting.

7; however, values

mit of distance for

Mouse provides access to all of the mouse services. Most of the important may be occasions when you need a mouse capability that we have not ones are provided as a simplified call with Graphics Workshop.

For instance, if you want to know how many times the ShowCursor routine needs to be called to make the mouse visible, this information is stored in Using mouse service 21, you can find the size of the mouse driver's environment. Then, using mouse service 22, you can retrieve the mouse environment. Since the count used by HideCursor and ShowCursor is not always in the same place you would have to save the mouse environment, make a call to HideCursor (which will change the count) and then retrieve a second copy of the environment. Compare the two environments until you find a byte which is different. During the session, this byte will always be the location of the count and you can retrieve the environment at any time to get the count. This complete example is contained in two routines in Graphics Workshop: GetCount-An example of using these routines n context is shown in the example program DEMOMOUS.BAS. Location% and GetMouseCount%. the mouse's environment.

See Alco.

GetCountLocation%, GetMouseCount%

Assembly Routines

ShowCursor

SetCursor

Assembler subroutine contained in GW.LIB

■ Purpose:

SetCursor provides a simple way to set a new location for the mouse cursor.

■ Syntax:

CALL SetCursor(X%, Y%)

Syntax:

CALL ShowCursor

ShowCursor turns or currently visible, Sh

visible.

Where:

tively. X% and Y% represent the new horizontal and vertical positions respec-

Comments:

Assembly Routines

The valid X and Y coordinates you specify will depend on the current screen mode. For example, on a CGA graphics screen 1, the acceptable range would be between 0 and 319 for X%, and 0 to 199 for Y%.

■ See Also:

HideCursor

For more information routine HideCursor.

Comments:

CHARGELL SUFFINALE DAG.

ShowCursor

Stred in ONLIB

Assembler Subroutine contained in GW.LIB

Purpose:

ShowCursor turns on the mouse cursor, making it visible. If the cursor is currently visible, ShowCursor does nothing, and leaves the mouse cursor visible.

Syntax:

tical positions respec-

CALL ShowCursor

Comments:

For more information see the comments that accompany the companion routine HideCursor.

Assembly Routines

See Also:

Pend on the current een 1, the acceptable 1199 for Y.R.

HideCursor

CRESCENT SOFTWARE, INC.

NT SOFTWARE BYC

3 - 117

Assembly Routines

Section 5:

and P.D.Q. Routines from QuickPak Professional

reason for adding these routines is that they are used by the subroutines VertMenuG and PullDownG to manage information. We have included several subroutines and functions from QuickPak Professional. These routines are described in this section. The only

Syntax:

Active = AltKeyA

Where:

Active receives -1 if

for this function. of all your programs

the BASIC NOT of the AltKey% is designed

15 AltKey% THE

IF NOT ATTREY

Because AltKey% h

Comments:

before it can be used

в Ригрозе:

Alikey's reports if

AltKey%

Assembly Routines

3 - 118

CRESCENT SOFTWARE, INC.

CHECKETT SUFFRANCE INC.

AltKey%

Sonal

Assembler function contained in GW.LIB

Purpose:

AltKey% reports if the Alt key is currently depressed.

■ Syntax:

is section. Orientals set by the submittings

Active = AltKey%

■ Where:

Active receives -1 if the Alt key is currently down, or 0 if it is not.

Comments:

Because AltKey% has been designed as a function, it must be declared before it can be used. Including the file GWDECL.BAS at the beginning of all your programs will avoid any problems, as it contains a declaration for this function.

AltKey% is designed to return -1 for a true value to also allow the use of the BASIC NOT operator:

IF AltKey% THEN .

ŗ

IF NOT AltKey% THEN

Assembly Routines

FindLast%

function contained in GW.LIB Assembler

purpose;

HeroThere's will re

cules graphic suppor

HercThere%

Purpose:

FindLast% scans a conventional (not fixed-length) string array backwards looking for the last non-blank element.

Syntax:

NumEls% = UBOUND(Array\$)
Last = FindLast%(BYVAL VARPTR(Array\$(NumEls%)), NumEls%)

Where:

Loaded will be set to

is resident, or 0 if it

Because HercThere?

Comments:

for this function.

of all your programs etore it can be used

Note that using this

Syntax:

Losdeds = HercTher

Where:

Assembly Routines

sioned. NumEls% is the number of elements to which Array\$() has been dimen-

Last receives the number of the last element that is not empty

Comments:

of all your programs will avoid any problems, as it contains a declaration before it can be used. Including the file GWDECL.BAS at the beginning for this function. Because FindLast % has been designed as a function, it must be declared

elements is use blank entries should the array be dimensioned larger than the number of vertical scrolling menu. The physical array can be larger than the number of elements used. It would be inappropriate for VertMenuG to display VertMenuG uses this routine to find the actual number of elements in the

> OBHERC.COM necessary. The

Even though Quink

CHARGE LANGE LANGE

See Also: will detect if either OBF MSHERC.COM when

QuickBASIC 4.0 come

of HercThere %. Support program, MSH display is installed in the The Graphics Worksho Function Call" error. the SCREEN 3 statem MSHERC.COM, must adapter, a special

contains the routines no

Jurean?

HercThere%

Zaned in CWIIB

String arm backwark

Assembler function contained in GW.LIB

Purpose:

HercThere% will report if the QBHERC.COM or MSHERC.COM Hercules graphic support program has been loaded into memory.

Syntax:

ME (SA)

Loaded% = HercThere%

Where:

ws/) has been dimen.

Loaded will be set to -1 if QBHERC.COM or MSHERC.COM program is resident, or 0 if it is not.

Assembly Routines

Comments:

not empty.

Because HercThere% has been designed as a function, it must be declared before it can be used. Including the file GWDECL.BAS at the beginning of all your programs will avoid any problems, as it contains a declaration for this function.

> AS at the beginning ontains a declaration

Note that using this method for doing Hercules graphics is not entirely necessary. The Graphics Workshop routine Screen3 sets the computer in of loading QBHERC. COM or MSHERC. COM. See the routine Screen3 for details. the Hercules high-resolution mode without the need

> r of elements in the ger than the number t.MenuG to display than the number of

MSHERC. COM, must be run first. If this is not done, attempting to use the SCREEN 3 statement to enter graphics mode will cause an "Illegal Even though QuickBASIC supports graphics using a Hercules display special TSR (terminate and stay resident) program, Function Call" error. The Graphics Workshop MultMonitor function will tell you if a Hercules display is installed in the host PC, but it does not detect if the necessary support program, MSHERC.COM, has been loaded. This is the purpose of HercThere%. QuickBASIC 4.0 comes with a program named QBHERC.COM that contains the routines necessary for Hercules graphics. It was renamed to MSHERC.COM when QuickBASIC 4.5 was introduced. HercThere will detect if either QBHERC.COM or MSHERC.COM is loaded.

See Also:

Screen 2

CRESCENT SOFTWARE, INC.

3 - 121

InStat%

function contained in GW.LIB Assembler

I Purpose:

space when you cor works without using PDQTimer& is a be

load its floating-poil

PDQTImer&

Purpose:

uses this function to avoid drawing the shadow effect if the user has already pressed an additional keystroke. InStat % returns the number of characters that are currently pending in the keyboard buffer, without removing them. The PullDownG menu system

Syntax:

Count% = InStat%

Where:

Where:

in the PC's internal PDQTimer& return

Comment:

Because PDQTimer&

Syntax:

NunTicks& = POOTin

Assembly Routines

any characters are pending, PullDownG will delay drawing the shadow Count% receives the number of characters that are currently pending. If

Comments:

for this function. of all your programs will avoid any problems, as it contains a declaration before it can be used. Including the file GWDECL.BAS at the beginning Because InStat% has been designed as a function, it must be declared

example would be when simulating multi-tasking in a BASIC program. present but do not want to remove it from the keyboard buffer. A good InStat% is very valuable in situations where you need to see if a key is

> oode tragment b There are 18 to for this function before it can be used. of all your program

WILLE Marticless > PI

CHECKLE WELLING W.

PDQTimer&

a ed in GWIIB

Assembler function contained in GW.LIB

Purpose:

Downs mem sing in the lift of the last new system

works without using floating-point arithmetic, which will save you code space when you compile your programs, since BASIC will not need to PDQTimer& is a better replacement for the BASIC TIMER statement. It load its floating-point math libraries.

■ Syntax:

NumTicks& = PDQTimer&

■ Where:

drawing the shadow

PDQTimer & returns a value of the number of clock ticks which occurred in the PC's internal clock since the last reset or overflow.

Comment:

i must be declared AAS at the beginning mains a declaration

Because PDQTimer& has been designed as a function, it must be declared before it can be used. Including the file GWDECL.BAS at the beginning of all your programs will avoid any problems, as it contains a declaration for this function. There are 18 ticks per second, so to delay 3 seconds wait 54 ticks. The code fragment below shows how to wait 3 seconds.

of to see if a key is and buffer. A good BASIC program.

NumTicks& = PDQTimer& + 54 WHILE NumTicks& > PDQTimer&: WEND

Assembly Routines



BASIC Routines

Chapter 4: BASIC Routines

BASIC Routines ON SHIMMEN MANAGED in addition, some o in addition, some o variables described variables described Chapter 1. The ben DisplayBox which di operator. The XOR (statement to draw a b not necessary with the programming with the programming or The most common warmens is Window poutines is Window coordinates of the up covered. This works to not necessary to mai you are maintaining HILE INERT = "": WE on the screen. be called a second in GWindow.XIX = 1 GWindow.XIX = 43 GWindow.XIX = 432 GWindow.YIX = 340 DIN GWindow 7-4 INLESO

This chapter contains all the routines which are written in BASIC. These routines are used either by the demo programs, or by one of the other BASIC routines to accomplish a task with respect to the graphics screen. Many of the routines provided are here to show how many calls to one of the Graphics Manipulation assembly routines, like the GMoveIVE routine, can be used to provide animation.

In addition, some of the BASIC routines take advantage of the TYPE variables described in the "Different Coordinate Systems" section of Chapter 1. The benefit of using these TYPE variables is apparent when you are maintaining information about multiple window regions, and it is not necessary to maintain four sets of variables. A general overview of programming with these window description variables will be given here. routines is Window1. Window1 describes a region by giving the pixel coordinates of the upper-left corner of a region and for the lower-right The most common variable type used in the Graphics Workshop BASIC corner. This works exactly like specifying a region for the BASIC LINE statement to draw a box. For this example we will use the BASIC routine DisplayBox which displays a box on the screen using the XOR logical operator. The XOR operator is used so that the DisplayBox routine can be called a second time to remove the box and restore the original image on the screen.

'Makes all variables integers	'defines a local TYPE variable	'Upper-left X coordinate 'Upper-left Y coordinate 'Lower-right X coordinate 'Lower-right Y coordinate	'draw the box using XOR	'wait for a keystroke	'remove the box
DEFINT A-Z	DIM GWWindow as Windowl	GWV indow.X1% = 51 GWW indow.Y1% = 43 GWW indow.X2% = 432 GWW indow.Y2% = 340	CALL DisplayBox(GWWindow) 'draw the box using XOR	WHILE INKEY\$ = "": WEND	CALL DisplayBox(GWWindow) 'remove the box

BASIC Routines

CircleBAS

contained in CIRCBAS.BAS **BASIC** subroutine

a purpose:

Copylmage

Purpose:

CircleBAS is a BASIC routine which not only explains the algorithm for drawing circles, but also provides a method for drawing circles on a Hercules screen by using the DrawPointH routine to plot pixels. This routine, as shipped, works with the EGA and VGA screen modes. Comments in the routine show how to modify it for Hercules.

Syntax:

CALL CircleBAS (XCenter%, YCenter%, Radius%, CircleColor%, XAspect%, YAspect%)

Where:

LRCol% and LRL ULCol% and ULLin

region. Together the

ollow the mixed coo

Syntax:

CALL CopyImage(UL

physically move the Copylmage copies

location on the scr

Where:

XCenter% and YCenter% define the center of the circle on the screen.

80 and line values ra

the upper-left corner

DestCol% and Dest

The code fragr Comments: Radius% is the radius of the circle in pixels

CircleColor% is the color used to draw the circle.

XAspect% and YAspect% are used to draw ellipses.

BASIC Routines

Comments:

for drawing circles and lines to the Hercules graphics screen. In future versions of Graphics Workshop, there will be assembly routines using the existing pixel setting routines. An assembly routine, CircleVE, exists and is much faster for drawing circles to EGA and VGA screens. This routine is designed to allow Hercules graphics screens to draw circles

See Also:

DrawPointH, LineBAS

See Also:

example program, For an example of hos

CALL Copylinage (1. SCREEN 12 sets the

GMovel VE

CHARLE WELLINE

Copylmage

ORCONSBAS

ain the algorithm for to pot pixels. This drawing circles on a VGA Sureen modes.

BASIC subroutine contained in COPYIMAG.BAS

Purpose:

Copylmage copies a specified block of the graphics screen to another the GMove1VE routine to This routine uses physically move the image. location on the screen.

Syntax:

, Hercules.

ecolory.

CALL CopyImage(ULCol, ULLine, LRCol, LRLine, DestCol, DestLine)

Where:

ULCol% and ULLine% specify the upper-left corner of the source region. LRCol% and LRLine% specify the lower-right corner of the source region. Together they specify the region to be moved. These coordinates follow the mixed coordinate system where column values range from 1 to 80 and line values range from 0 to 479 on a VGA monitor.

ircle on the screen,

DestCol% and DestLine% specify a mixed coordinate value which tells the upper-left corner of the image's final resting place.

Comments:

BASIC Routines

The code fragment below demonstrates copying an image 3 columns wide by 20 lines directly to the right of the original image.

DEFINT A-Z

and VGA screens. assembly routines

cens to draw circles

'\$INCLUDE: 'GWDECL.BAS' SCREEN 12'sets the monitor in VGA mode

CALL CopyImage (1, 0, 3, 19, 4, 0)

For an example of how to use this routine, see the DEMOMOVE.BAS example program.

See Also:

GMove1VE

DisplayBox

purpose:

DisplayBox draws a

with exclusive-orin

Digitize

BASIC subroutine contained in DIGITIZE.BAS

Purpose:

specified region on the screen. This routine simulates the way features, such as a person's face, are disguised on television. Digitize takes a graphics image and builds a lower resolution version of a

Syntax:

CALL Digitize(GWWindow AS Window1, Pixels%, Quick%)

Where:

defined in the inclu GWWindow is a B

with variables of the structure are X1%, Syntax:

CALL DisplayBox

screen.

Where:

defined in the include file GWDECL.BAS. The variables in this record structure are X1%, Y1%, X2%, and Y2%. An example of programming with variables of this kind is shown at the beginning of this chapter. GWWindow is a BASIC TYPE structure. The structure Window1 is

Pixels% is the size of the boxes created.

to merely use the color which is in the center of the box. the routine not to do any calculation of the overall color of the region, but Quick% is a boolean variable which, if set to any non-zero value, tells

BASIC Routines

Comments:

example program. A demonstration of this routine can be found in the DEMODIGI.BAS

example progr

A demonstrațio

This routine uses Lin

Comments:

CRESCENT SOFTWARE, INC.

ON SAMUASS MADARD

ASIC Subroutine

DisplayBox

BASIC subroutine contained in XORBOX.BAS

Purpose:

Ext. ion version of a des the war features,

DisplayBox draws a box outlining the region specified. The box is drawn with exclusive-oring (XOR operator) to retain images already on the screen.

Syntax:

CALL DisplayBox(GWWindow AS Window1)

Where:

nothre Windows is Tables in this record ple of programming

of this chapter.

GWWindow is a BASIC TYPE structure. The structure Window1 is defined in the include file GWDECL.BAS. The variables in this record structure are X1%, Y1%, X2%, and Y2%. An example of programming with variables of this kind is shown at the beginning of this chapter.

Comments:

On-zero value, tells or of the region, but

This routine uses LineVE to draw the lines quickly, using the XOR ability of that routine. A demonstration of this routine can be found in the DEMOBOX.BAS example program.

BASIC Routines

T SOFTWARE, INC.

DisplayPCXFI

Purpose:

to the specified vid DisplayPCXFile wil

loading the image at

DisplayBoxFill

contained in XORBOX.BAS **BASIC** subroutine

Purpose:

the screen. is drawn with exclusive-oring (XOR operator) to retain images already on DisplayBoxFill draws a filled box covering the region specified. The box

Syntax:

CALL DisplayBoxFill(GWWindow AS Window1)

Where:

■ Syntax:

CALL DisplayPCXFi

Where:

GWWindow is a BASIC TYPE structure. The structure Window1 is defined in the include file GWDECL.BAS. The variables in this record structure are X1%, Y1%, X2%, and Y2%. An example of programming with variables of this kind is shown at the beginning of this chapter.

Comments:

BASIC Routines

of that routine. This routine uses LineVE to draw the lines quickly, using the XOR ability

a value of 0 here wil

Comments:

routine, your be already in the The screen mode is have redirected the v VideoPage% tells it .PCX extension is n Filenames gives the

example program. A demonstration of this routine can be found in the DEMOBOX.BAS

See Also:

This routine also calls

HandlePCXPalette, W

■ 4-6

CRESCENT SOFTWARE, INC.

HSIC subroutine T YORBOY BAS

DisplayPCXFile

BASIC subroutine contained in DISPLPCX.BAS

Specified. The hox An images already on

to the specified video page. This routine handles the entire process of DisplayPCXFile will load the PCX file that is passed to it as a parameter loading the image and making the appropriate adjustments to the palette.

CALL DisplayPCXFile(Filename\$, VideoPage%) Syntax:

Where:

ructure Window1 is iables in this record role of programming

of this chapter.

Filename\$ gives the name of the .PCX graphics file to be loaded. If a PCX extension is not given it will be appended.

have redirected the visual page by using the BASIC SCREEN statement, a value of 0 here will display the image to the currently visible page. VideoPage% tells it which video page to draw the image on. Unless you

Comments:

ing the XOR ability

DEMOBOX.BAS

The screen mode is not set by this routine if the monitor is presumed to be already in the proper screen mode. It is assumed that when calling this Comments in the code show how this routine can always modify the screen routine, your program will already be using a graphics screen mode.

This routine also calls HandlePCXPalette and WhichPCXScreen to handle interpreting the PCX header information.

HandlePCXPalette, WhichPCXScreen

DisplayPCXFile2

contained in DISPLPC2.BAS BASIC subroutine

purpose:

screen to the visible Doseguel is used i Dosegue

Purpose:

adjustments to the palette. handles the entire process of loading the image and making the appropriate DisplayPCXFile2 will load the PCX file that is passed to it as a parameter at the specified coordinates on the specified video page. This routine

CALL DisplayPCXFile2 (Filename\$, VideoPage%, LineStart%, ColStart%)

Where:

SubType% selects

Syntax:

Where:

Filename\$ gives the name of the .PCX graphics file to be loaded. If a .PCX extension is not given it will be appended.

a value of 0 here will display the image to the currently visible page. have redirected the visual page by using the BASIC SCREEN statement, VideoPage% tells it which video page to draw the image on. Unless you

SegueColor% is the

EGA or VGA high-resolution image, no positioning will take place. values are 0, no positioning will take place. Also if the image is not an LineStart% and ColStart% are in the Mixed Coordinate System. If both

BASIC Routines

Comments:

Comments in the code show how this routine can always modify the screen routine, your program will already be using a graphics screen mode. be already in the proper screen mode. It is assumed that when calling this The screen mode is not set by this routine if the monitor is presumed to

■ See Also:

makes the entire in

Comments:

of the Segue transfe Delay% tells the

DoSegue2, DoSegue3

the image has been opened with OpenPCXFile%. This routine calls upon PositionPCXVE to position the PCX image after

interpreting the PCX header information. This routine also calls HandlePCXPalette and WhichPCXScreen to handle

See Also:

HandlePCXPalette, WhichPCXScreen

DESCRIPTION ASSOCIATION AND VALUE OF THE PARTY OF THE PAR

DSC Subroune DSC PRINTING

DoSegue1

BASIC subroutine contained in SEGUE1.BAS

■ Purpose:

to i as commented in the second

DoSeguel is used by QuickSegue to transfer the background graphics screen to the visible graphics screen.

■ Syntax:

CALL DoSegue1(SubType%, SegueColor%, Delay%)

■ Where:

artz, ColStarts)

to be loaded. If a

ege on. Unless you CREEN statement,

ly visible page.

ite System. If both the image is not an

ill take place.

SubType% selects a subtype for the style used in DoSegue1.

SubType% DESCRIPTION

1 Transfers entire screen instantaneously.
2 Paints entire screen in SegueColor %.

SegueColor% is the color to place on the screen when the subtype is 2.

Delay% tells the routine how many milliseconds to delay after each step of the Segue transfer.

BASIC Routines

Comments:

This transition cannot be described using a picture. This routine simply makes the entire image appear in one shot.

■ See Also:

ics screen mode. modify the screen

Or is presumed to when calling this Screen to handle

PCX image after

DoSegue2, DoSegue3, DoSegue4, DoSegue5, DoSegue6.

CRESCENT SOFTWARE, INC.

TSOFTWARE, BIC

4 - 9

DoSegue2

contained in SEGUE2.BAS **BASIC** subroutine

Purpose:

screen to the visible graphics screen. DoSegue2 is used by QuickSegue to transfer the background graphics

Syntax:

CALL DoSegue3(SubT

purpose:

screen to the visible DoSegue3 is used b Dosegues

Syntax:

CALL DoSegue2(SubType%, SegueColor%, Delay%)

Where:

delayed iteration. the value in the variable SubType%, the larger the chuck displayed at each SubType% selects a subtype for the style used in DoSegue2. The larger

chunks to fill the scn SubType% selects a values for the variable

1,3,5,7,9,11

_	SubType%	SubType% DESCRIPTION
	1,3,5,7	Paints the screen in the fashion shown below.
	2,4,6,8	Brings in the background image in the fashion shown below
		shown below.

SegueColor% is the color to place on the screen when the subtype is an odd value.

BASIC Routines

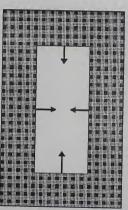
of the Segue transfer. Delay% tells the routine how many milliseconds to delay after each step

Comments:

DoSegue2 routine. The picture below shows the direction of the screen coverage for the

of the Segue transfer.

SegueColor% Each of the higher v



See Also:

DoSegue1, DoSegue3, DoSegue4, DoSegue5, DoSegue6

SKSC Subsouther

DoSegue3

BASIC subroutine contained in SEGUE3.BAS

Purpose:

Kerr and Erephiss

DoSegue3 is used by QuickSegue to transfer the background graphics screen to the visible graphics screen.

■ Syntax:

CALL DoSegue3(SubType%, SegueColor%, Delay%)

■ Where:

OSeguez. The larger uck displayed at each

SubType% selects a subtype for the style used in DoSegue3. The larger values for the variable SubType% select different random patterns for the chunks to fill the screen.

SubType% DESCRIPTION 1,3,5,7,9,11 Paints the screen in the fashion s

1,3,5,7,9,11 Paints the screen in the fashion shown below. 2,4,6,8,10,12 Brings in the background image in the fashion shown below.

Each of the higher values produces a different random display.

in the subtype is an

below. fashion lay after each step

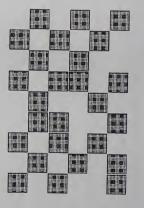
coverage for the

SegueColor% is the color to place on the screen when the subtype is an odd value. Delay% tells the routine how many milliseconds to delay after each step of the Segue transfer.

BASIC Routines

Comments:

and others are still missing. The picture below shows how the screen might look part way through the execution of this routine. Some of the blocks in the image have appeared



Where:

SuhType% selects a

delayed iteration.

SubType% DI

the value in the varia

syntax:

(4) DoSegue4(Sub)

purpose:

Dosegue4 is used t

screen to the visible

DoSegue4

See Also:

BASIC Routines

DoSegue1, DoSegue2, DoSegue4, DoSegue5, DoSegue6.

ON BRYMAIN MASKED

Downer Downer T

See Also:

DoSegued routine The picture below sho of the Segue transfer.

Delay% tells the rout

odd value. SegueColor%

For higher values, 12 The dots mean that

2,4,6,8,...

Comments;

The Control of the Co A Taken Track the

DoSegue4

BASIC subroutine contained in SEGUE4.BAS

Purpose:

DoSegue4 is used by QuickSegue to transfer the background graphics screen to the visible graphics screen.

Syntax:

CALL DoSegue4(SubType%, SegueColor%, Delay%)

Where:

SubType% selects a subtype for the style used in DoSegue4. The larger the value in the variable SubType%, the larger the chuck displayed at each delayed iteration.

DESCRIPTION SubType%

Paints the screen in the fashion shown below. 1,3,5,7,...

2,4,6,8,...

Brings in the background image in the fashion shown below.

The dots mean that this routine can use higher values if you so choose. For higher values, the blocks will become very large.

BASIC Routines

SegueColor% is the color to place on the screen when the subtype is an Delay% tells the routine how many milliseconds to delay after each step odd value.

Comments:

of the Segue transfer.

The picture below shows the direction of the screen coverage for the DoSegue4 routine.



See Also:

DoSegue1, DoSegue2, DoSegue3, DoSegue5, DoSegue6.

DoSegue5

contained in SEGUE5.BAS **BASIC** subroutine

Purpose:

screen to the visible Dosegues is used b

Purpose:

screen to the visible graphics screen. DoSegue5 is used by QuickSegue to transfer the background graphics

CALL DoSegue5(SubType%, SegueColor%, Delay%)

Where:

SubType% selects a

delayed iteration. the value in the variat Syntax:

CHIL DoSegue6(Sub)

Where:

SubType% selects a subtype for the style used in DoSegue5. The larger the value in the variable SubType%, the larger the number of lines growing for each iteration.

SubType% DESCRIPTION

1,3,5,7 2,4,6,8 Brings in the background image in the fashion shown below.3 Paints the screen in the fashion shown below.

odd value. SegueColor% is the color to place on the screen when the subtype is an

SegueColor% is the

Delay% tells th.

of the Segue transier

BASIC Routines

of the Segue transfer. Delay% tells the routine how many milliseconds to delay after each step

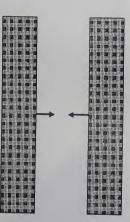
Comments:

DoSegue5 routine. The picture below shows the direction of the screen coverage for the

The picture below sho

DoSegues routine

Comments:



See Also:

DoSegue1, DoSegue2, DoSegue3, DoSegue4, DoSegue6

See Also:

Dossguel, Dossegue2, D

BASCS Pouline SEQUESTRA

DoSegue6

BASIC subroutine contained in SEGUE6.BAS

Purpose:

Seround Staphiss

DoSegue6 is used by QuickSegue to transfer the background graphics screen to the visible graphics screen.

■ Syntax:

CALL DoSegue6(SubType%, SegueColor%, Delay%)

■ Where:

OSegues. The karger mber of fines growing

SubType% selects a subtype for the style used in DoSegue6. The larger the value in the variable SubType%, the larger the chuck displayed at each delayed iteration.

7	Paints the screen in the fashion shown below.	Brings in the background image in the fashio
SubType% DESCRIPTION	Paints the screen	Brings in the bac
SubType%	1,3,5,7	2,4,6,8

SegueColor% is the color to place on the screen when the subtype is an odd value.

en the subtype is an

fashion

'lay after each step

BASIC Routines

Delay% tells the routine how many milliseconds to delay after each step of the Segue transfer.

Comments:

coverage for the

The picture below shows the direction of the screen coverage for the DoSegue6 routine.



See Also:

DoSegue1, DoSegue2, DoSegue3, DoSegue4, DoSegue5.

T SOFTWARE, DV.

DrawCursor

DrawCursor draws a

operation at the spec

Draw3DButton

BASIC subroutine contained in BUTTON.BAS

Purpose:

location. Draw3DButton draws a three-dimensional button on screen at the specified

Syntax:

CALL Draw3DButton(XPos%, YPos%, Wdth%, Height%, ThirdDimension%, Colr%)

Where:

XPos% and YPos% specify the upper-left corner of the three-dimensional

XPos% and YPos%

CALL DrawCursor(Xr

Wdth% and Height% specify the size of the button in pixels.

button its three-dimensional look. ThirdDimension% specifies the number of pixels involved in giving the

0n0r0ff% specifies Height% specifies th Width % specifies the cursor. These position

A value other than ze

Timing% holds the

color. a bright white highlight, and the bottom of the button uses a black shadow Colr% is the color for the face of the button. The top of the button uses

BASIC Routines

Comments:

colors. The routine could be modified to give different highlight and shadow

See Also:

demonstrated in the ex-DEMOCURS.BAS. T A demonstration of D the Exclusive-ORed ct

DrawCursor uses the Comments: complete cycle cursor on and h

GEditor.

ON SHIRMAN HADORD

DrawCursor

BASC Subsouline
BUTTO Nation

BASIC subroutine contained in CURSOR.BAS

Purpose:

Street at the specified

DrawCursor draws a graphics cursor on the screen using the XOR logical operation at the specified location.

■ Syntax:

CALL DrawCursor(XPos%, YPos%, Wdth%, Height%, OnOrOff%, Timing%)

■ Where:

XPos% and YPos% specify the upper-left coordinate to start drawing the cursor. These positions are absolute pixel locations on the screen.

Wdth% specifies the horizontal width of the cursor in pixels.

Height% specifies the height of the cursor.

volved in giving the

in pixels.

p of the button uses

OnOrOff% specifies whether the cursor is currently visible on the screen. A value other than zero means that the cursor is visible. Timing% holds the time (in 1/18ths of a second) between blinking the A value of 9 would produce a cursor on and blinking the cursor off. complete cycle once every second.

BASIC Routines

Comments:

Hight and shadow

DrawCursor uses the LineVE routine from Graphics Workshop to draw the Exclusive-ORed cursor on the screen. A demonstration of DrawCursor can be found in the example program DEMOCURS.BAS. The GEditor routine also uses DrawCursor, and is demonstrated in the example program DEMOEDIT.BAS.

■ See Also

GEditor

DrawText

contained in DRAWTEXT.BAS BASIC subroutine

п Ригрозе:

to the full scale. window size and cal FullZoom zooms in FullZoom

Purpose:

GraphPak Professional. DrawText is used to draw a text string using the fonts available with

Syntax:

CALL DrawText(X%, Y%, Text\$, Angle%, Colr%, SizeMultiplier#

Where:

the first character to be drawn The variables X% and Y% are pixel positions of the upper-left corner of

Text\$ is a string of the phrase to be drawn.

with variables of this structure are X1%, 1 defined in the includ GWWindow is a B

Where:

CALL FullZoon(GA)

Comments:

DEMOZOOM.BAS The FullZoom rou

An angle of 0 draws text straight across the screen. Angle% is the angle specified in degrees at which the text is to be drawn.

activate the shadow effect. have shadows drawn underneath them. Adding 128 to the color will Colr% is the color of the text. Both DrawText and StepText phrases can

BASIC Routines

of the font. the original definition of the font, 2# twice as big, .75 = 3/4 the size SizeMultiplier# is the size of the font. For example, 1# = same size as

Comments:

about the GPDat%() array. modifying the variable GPDat%(14). See Appendix C for information The color for the shadow effect mentioned above can be changed by

The GraphPak fonts are used throughout the demonstration programs, and particularly in the example program DEMOFONT.BAS.

See Also:

GetWidth%, StepText

ON STRUMENT MEDICAL

FullZoom

DEWITERIES

BASIC subroutine contained in ZOOM.BAS

Purpose:

e fints are lable with

Full Zoom zooms in on a portion of the screen. The algorithm takes the window size and calculates the proportions necessary to zoom the image to the full scale.

■ Syntax:

CALL FullZoom(GWWindow AS Window1)

W/house

e upper-left corner of

GWWindow is a BASIC TYPE structure. The structure Window1 is defined in the include file GWDECL.BAS. The variables in this record structure are X1%, X1%, X2%, and Y2%. An example of programming with variables of this kind is shown at the beginning of this chapter.

Comments:

e text is to be drawn.

tepText phrases can 8 to the color will

1# = same size as ... 75 = 3/4 the size

C for information

in be changed by

ion programs, and

The FullZoom routine is demonstrated in the example program DEMOZOOM.BAS.

BASIC Routines

CRESCENT SOFTWARE, INC.

4-19

CetCountLoca

GetCountLocation 8

GEditor

contained in GEDITOR.BAS **BASIC** subroutine

Purpose:

high-resolution screen modes. GEditor is a graphics mode text input routine that also allows editing an existing string. GEditor is designed for use with the VGA and EGA

CALL GEditor(Edit\$, LeftCol%, KeyCode%, TxtPos%)

Syntax:

CountPosition% = 1

Because GetCountLo Comments:

eclared before it car

a call to Show/Cursor not the mouse cursor

been rumed off is lo vironment of where

Where:

the correct maximum length (see below). Edit\$ is the string being entered or edited, which must be pre-assigned to

LeftCo1% is the column number used to place the cursor when editing the string. LeftCo1% is maintained in this fashion to take advantage of the re-entrant abilities of the GEditor routine.

code of the last character typed, or it returns the negative value of an KeyCode% indicates how editing was terminated. It returns the ASCII extended character if an extended character was typed.

BASIC Routines

TxtPos% is the current position of the cursor within the edited string.

Comments:

would specify an Edit\$ with 16 spaces as in the example below: If you want to restrict the length of the input to 16 characters, then you

> screen mode, as the mo We suggest that your

GetMouseCount* ment which will give This function tells o declaration for this h neginning of all your

Edit\$ = SPACE\$(16)

CALL GEditor(Edit\$, LeftCol%, KeyCode%, TxtPos%)

as in the example below: If Edit\$ already contains information, you would pad the string with blanks

Edit\$ = Edit\$ + SPACE\$(16 - LEN(Edit\$))

See Also;

program DEMOMOUS A complete demonstra take time and valuable modes. We also sugge

GetMouseCounts

program DEMOEDIT.BAS. A complete demonstration of the GEditor routine is given in the example

while in graphics mode. The GEditor routine uses the DrawCursor routine to give the user a cursor

See Also:

DrawCursor

GetCountLocation%

BASIC function contained in MOUSECNT.BAS

Purpose

the NGA and EGA

GetCountLocation% obtains the location within the mouse driver's environment of where the count of how many times the mouse cursor has been turned off is located. This can be useful in telling you whether or not the mouse cursor is currently visible, or telling you how many times a call to ShowCursor is required to make the cursor visible once again.

Syntax:

ist be pre-assigned to

CountPosition% = GetCountLocation%

Comments:

Sor when editing the ke advantage of the

t returns the ASCII egative value of an

Because GetCountLocation% has been designed as a function, it must be declared before it can be used. Including the file GWDECL BAS at the beginning of all your programs will avoid any problems, as it contains a declaration for this function.

GetMouseCount% is required to actually determine whether or not the This function tells only the location of the variable in the mouse environ-A call to the function ment which will give you this information. mouse cursor is visible.

BASIC Routines

We suggest that you use this routine immediately after setting your graphics screen mode, as the mouse environment does change after changing screen modes. We also suggest you execute this function just once, as it does take time and valuable data space to perform this operation.

aracters, then you

le below:

he edited string.

A complete demonstration of this routine can be found in the example program DEMOMOUS.BAS.

See Also:

string with blanks

en in the example

the user a cursor

GetMouseCount%

GetMouseCount%

contained in MOUSECNT.BAS **BASIC** function

purpose:

the vector font width GetOutlineWidth®

Syntax:

Width% = GetOutlin

Where:

Texts is a string wh

Because GetOutline

eclared before it co

Comments:

beginning of all your

declaration for this t

GetOutlineW

П Purpose:

GetMouseCount% tells you how many times the HideCursor routine has been called. If the mouse is currently visible then this routine will return a zero for the number of times the HideCursor routine has been called.

MouseCount% = GetMouseCount%(CountPosition%)

Where:

CountPosition% is the position of the count variable within the mouse driver's environment. To get the CountPosition%, a call to the function GetCountLocation% is made as shown in the example below:

CountPosition% = GetCountLocation%

Comments:

BASIC Routines

Because GetMouseCount% has been designed as a function, it must be declared before it can be used. Including the file GWDECL.BAS at the beginning of all your programs will avoid any problems, as it contains a declaration for this function.

The width is for a fo

other size, simply m

To set up the mouse driver and check to see if the mouse cursor is active, execute the following code:

variable GPDat%(42 Centering a Graj the actual number

GALL OutLineText(Starts = (600m) 49

'\$INCLUDE: 'GWDECL BAS'
SCREEN 9 'optional, as GetMouseCount% works DEFINT A-Z 'Good practice for all programs in all screen modes

CALL InitMouse(There%) 'Initialize the mouse IF NOT There% THEN PRINT "No Mouse Driver": END

See Also:

to start drawing the su the total width of the This takes half of the

OutlineText

IF GetMouseCount%(CountPostion%) = 0 THEN
PRINT "Mouse visible" CountPosition% = GetCountLocation% PRINT "Mouse not visible"

program DEMOMOUS.BAS A complete demonstration of the routine can be found in the example

See Also:

GetCountLocation%

ON STRUCTURE LANDS

GetOutlineWidth%

OUSECNTBAS

ideCursor nume has Le courie will ream

ine has been called.

BASIC function contained in OUTLTEXT.BAS

Purpose:

GetOutlineWidth% determines the width in pixels of a text string using the vector font widths of the currently loaded vector font.

Width% = GetOutlineWidth%(Text\$)

Where:

le within the mouse

cal to the function

e below.

unction, it must be VDECL. BAS at the ns, as it contains a

Text\$ is a string which holds the phrase to be drawn.

Comments:

Because GetOutlineWidth% has been designed as a function, it must be declared before it can be used. Including the file GWDECL, BAS at the beginning of all your programs will avoid any problems, as it contains a declaration for this function.

other size, simply multiply that size by the width of the text string to obtain The width is for a font size of 1. If you will be drawing the text at any the actual number of pixels that will be used.

BASIC Routines

Centering a Graphics Workshop font can be done by using the screen width variable GPDat%(43) and the value returned by this routine.

se cursor is active,

Start% = (GPDat%(43) - GetOutlineWidth%(Text\$)) \ 2
CALL OutLineText(Start%, Y%, Text\$, Angle%, Colr%, Wult%, Divid%)

This takes half of the screen size as being the center and subtracts half of the total width of the string in pixels. This will give you the X position to start drawing the string, in order to have it appear centered.

See Also:

OutlineText

in the example

GetWidth%

contained in DRAWTEXT.BAS **BASIC** function

Purpose:

GPaintBox changes

rectangular region o

GPaintBox

Purpose:

GetWidth determines the width in pixels of a text string using the font widths of the currently loaded GraphPak font.

Syntax:

Width% = GetWidth%(Text\$)

Where:

Text\$ is a string which holds the phrase to be drawn.

Where:

CALL GPaintBox(ULC

ULCol% and ULLi

Comments:

for this function. of all your programs will avoid any problems, as it contains a declaration before it can be used. Including the file GWDECL.BAS at the beginning Because GetWidth % has been designed as a function, it must be declared

-I, the value in GPD:

New Color % is the

OldColor% is the w

80 and line values ra follow the mixed coo LRCol% and LRLi

Together they specif

that is going to be used in order to obtain the width in pixels of the string you will actually be placing on the screen. The width is for a font size of 1. It must be multiplied by the font size

BASIC Routines

GPDat % (43) and the value returned by this routine. Centering a GraphPak font can be done by using the screen width variable

Start% = (GPDat%(43) - GetWidth%(Text\$)) \ 2 CALL DrawText(Start%, Y%, Text\$, Angle%, Colr%, TextSize#)

If you specify a -

a value you must let GPDat%() element 5

will use the color in t

to start drawing the string, in order to have it appear centered. the total width of the string in pixels. This takes half of the screen size as being the center, and subtracts half of This will give you the X position

See Also:

DrawText, StepText

See Also:

routine modifies only t MsgBoxG to display a GPaintBox is used by

PullbownG, VertMenu

ON SEWMAN MESSED

BASICiunction DRAITETTENS

GPaintBox

BASIC subroutine contained in GPAINTBX.BAS

Purpose:

tring using the four

GPaintBox changes all occurrences of one color to another color within a rectangular region of the screen.

Syntax:

CALL GPaintBox(ULCol, ULLine, LRCol, LRLine, OldColor, NewColor)

Where:

Together they specify the region to be re-colored. These coordinates ULCol% and ULLine% specify the upper-left corner of the region. LRCol% and LRLine% specify the lower-right corner of the region. follow the mixed coordinate system where column values range from 1 to 80 and line values range from 0 to 479 on a VGA monitor.

> it must be declared SAS at the beginning Ontains a declaration

OldColor% is the color you want to have changed. If OldColor is set to -1, the value in GPDat%(57) is used.

ied by the font size pixels of the string

NewColor% is the new color to replace the former color.

BASIC Routines

Comment:

reen width variable

GPDat%() element 57 holds the background color for the screen. This is a value you must set to specify the overall background color of the screen. If you specify a -1 for OldColor% and GPDat%(57) is zero, GPaintBox will use the color in the upper-left most corner of the region. GPaintBox is used by the menu routines PullDownG, VertMenuG, and MsgBoxG to display a shadow effect. The shadow effect produced by this routine modifies only the background color for the screen.

nd subtracts half of you the X position

intered.

PullDownG, VertMenuG, MsgBoxG

HandlePCXPalette

contained in PCXHEADR.BAS **BASIC** subroutine

purpose:

Interlude

Purpose:

sets the screens palette HandlePCXPalette takes the palette information from the PCX image and

Syntax:

CALL HandlePCXPalette(Array\$, WhichScreen%)

Where:

of the PCX header information is contained in Appendix A. Array\$ contains all the header information for the PCX file. A description

Where:

characters. Underst Text\$ is a string which

before calling this ro The font used by this Comments: Syntax:

CALL Interludes(Te

it can be added to ar

of your choice. This Interluded places a M

palette information to the various hardware configurations. WhichScreen% tells the routine the intended screen mode for the PCX HandlePCXPalette uses this to determine how to communicate the

Comments:

BASIC Routines

example program VIEWPCX.BAS, or in the routine DisplayPCXFile. To get a PCX header into Array\$, the routine OpenPCXFile% is used. An example of using the routine HandlePCXPalette can be found in the

standard palette. PCX files do not have palette information in them, and therefore use the This routine handles the different idiosyncracies of the PCX palette. Some

See Appendix B for more on palettes. 256,000. Note the VGA palette numbers are not necessarily sequential. The VGA has a maximum palette of 256 colors out of approximately modes. The EGA has a maximum palette of 16 colors out of 64 choices. For most cases, all of the palette information is for the EGA and VGA

See Also:

Because Graphics Wo

Text\$ = "Graph cs_ something like below to fit more text on the text for the

you are using many The fonts will size #

not take up much in th

SetOWFont, Interlude?

attempts to map the VGA's large palette into the EGA's limited palette. In the case of a VGA image being displayed on an EGA display, this routine

See Also:

OpenPCXFile%

CHANGE WATER

Interlude1

PCHEADREAS

BASIC subroutine contained in INTER1.BAS

the PCX impge and

Interlude1 places a Movie Director's Clicker on the screen with a message of your choice. This routine can be used by the QuickSegue program, or it can be added to any program to display a title.

Syntax:

CALL Interlude1(Text\$)

X file. A description

ndix A.

mode for the PCX to communicate the

Text\$ is a string which can have 1 or more phrases separated by underscore characters. Underscore characters tell the routine to start a new line.

Comments:

The font used by this routine can be specified using the SetGWFont routine before calling this routine. The fonts will size themselves automatically to fit inside the Clicker. If large a font for your tastes, add an equal amount of spaces to each side of the text for that line. This will force this routine into believing it needs to fit more text on the line and therefore create a smaller font. Try you are using many lines of text and this routine makes a line with too something like below:

can be found in the PCXFile% is used.

Display PCXFile.

CX palette. Some therefore use the

BASIC Routines

Made Text\$ = "Graphics_

te EGA and VGA

out of 64 choices. szarily sequential.

of approximately

Because Graphics Workshop uses a proportional font system, spaces do not take up much in the size of a font so you may want to add many of _Easy"

See Also:

SetGWFont, Interlude2.

splay, this routine

imited palette.

Interlude2

contained in INTER2.BAS **BASIC** subroutine

Purpose:

LineBAS demonstra

Purpose:

as being in the foreground. This routine adds motion to an otherwise static display. tions. The boxes move beneath all items on the screen which are marked Interlude2 moves rectangular boxes across the screen in random direc-

Syntax:

CALL Interlude2(TextColors%(), MaxBoxes%, Seconds%)

Where:

Coordinate pairs (x1

The LineColor% is

Syntax:

it to use the Hercule

uses the EGA and V screen mode by usin template for creating

picture. of the screen. You use this to identify which objects are part of your colors. Any items using these colors will be considered in the foreground TextColors%() is an array of colors which have been used as foreground creating its rectangular boxes. All other colors (0 - 15) will be valid for use by the routine in

one time. Use this to control the speed of the moving boxes. If there are moving so many boxes. too many boxes (i.e. 20) they will move slowly due to the overhead of MaxBoxes% is the maximum number of moving boxes on screen at any

BASIC Routines

Seconds% seconds. dependent upon the speed of any boxes left on screen at the end of Seconds% is the number of seconds for the routine to run. This will be

See Also:

routines for drawi screens. In fut ines using the This routine is des

CircleBAS, DrawPoin

Comments:

as five seconds might be added to the time this routine executes. could take as long as five seconds to cross the entire screen, so as many until all boxes have exited the screen before exiting. The routine may run for more seconds than were requested, as it waits The average box

others because they skip a line as they move across the screen. 80 pixels to 200 pixels in length. In addition some boxes move faster than The box sizes are random and range from 20 pixels to 40 pixels wide, and

See Also:

Interlude1.

ON BERNAME LICENSER

SSC suboutine O IN VIEW BAS

LineBAS

BASIC subroutine contained in LINEBAS.BAS

Purpose:

een in random dires. 20 which are marked to an otherwise static

LineBAS demonstrates the algorithm used for drawing lines, and gives a template for creating a line routine which will work with the Hercules screen mode by using DrawPointH to plot pixels. The routine, as shipped, uses the EGA and VGA screen modes, but has comments on how to change it to use the Hercules graphics screen.

Syntax:

CALL LineBAS (x1%, y1%, x2%, y2%, LineColor%)

used as foreground ed in the foreground ts are part of your se by the routine in

Coordinate pairs (x1%, y1%) and (x2%, y2%) are within the range of the

The LineColor% is the color of the line.

Comments:

es on screen at any boxes. If there are to the overhead of

lines using the existing pixel setting routines. An assembly routine, LineVE, exists and is much faster for drawing circles to EGA and VGA screens. In future versions of Graphics Workshop, there will be assembly routines for drawing lines and circles to the Hercules graphics screen. This routine is designed only to allow Hercules graphics screens to draw

BASIC Routines

en at the end of

CircleBAS, DrawPointH

Jested, as it waits The average box reen, so as many pixels wide, and move faster than

executes.

LoadFont

BASIC subroutine contained in DRAWTEXT.BAS

purpose:

LoadOutlineFont los

LoadOutlineFo

Purpose:

font definitions can be loaded at any point in time. LoadFont loads the specified GraphPak font file into a Font\$0 array for later use by the text drawing routines, DrawText and StepText. Multiple

Syntax:

CALL SetGPFont(1)
CALL LoadFont(FontFile\$)

Where:

FontFile\$ is a filename for one of the GraphPak style fonts. This string may include a path if desired.

■ Where

This string may inclu

FontFiles is a filena

Comment:

Syntax:

CALL LoadOut lineFo CALL SetGWFont(1) any point in time.

routines, Outline Tex specified into a Out

Comments:

list of other font files is located at the end of this chapter. filename passed to the routine. A standard font file is HELV12.GFN. A The file extension ".GFN" is assumed and automatically added to each

filename passed to it The file extension of

all available font file

BASIC Routines

with the font we are loading. All loaded fonts remain in memory. The font selected by SetGPFont will be the font used by the next calls to DrawText or StepText. The call to SetGPFont tells the LoadFont routine which number to identify

See Also:

SetGPFont, DrawText, StepText

■ See Also:

to identify with The call to Set

in memory, and can

SetGWFont, Outline?

CRESCENT SOFTWARE, INC.

CHARLE WAS THE THE

DEC. Schooling

LoadOutlineFont

BASIC subroutine contained in OUTLTEXT.BAS

■ Purpose

Septem Mulpie

specified into a OutlineFont\$() array for later use by the text drawing routines, OutlineText. Multiple vector font definitions can be loaded at LoadOutlineFont loads the specified Graphics Workshop Vector font file any point in time.

■ Syntax:

CALL SetGWFont(1)
CALL LoadOutlineFont(FontFile\$)

■ Whor

e fonts. This string

FontFile\$ is a filename for one of the Graphics Workshop Vector fonts. This string may include a path if desired.

Comment:

cally added to each HELV12.GFN. A

number to identify in memory. The the next calls to

The file extension of ".QFN" is automatically assumed and added to each filename passed to it. A standard outline font is HELV.QFN. A list of all available font files is located at the end of this chapter.

to identify with the Vector font we are loading. All loaded fonts remain in memory, and can be selected with the SetGWFont routine at any time. The call to SetGWFont tells the LoadOutlineFont routine which number

■ See Also:

SetGWFont, OutlineText

BASIC Routines

Lts2MenuG

The Las2MemuG rot

LtsMenuG

contained in LTSMENU.BAS **BASIC** subroutine

Purpose:

or by pressing a key that corresponds to the first letter of a choice. The LtsMenuG routine provides a single-line Lotus 1-2-3 "look alike" menu system. A list of choices is displayed horizontally on a single line. A choice can be made either by using the arrow keys and pressing Enter,

Syntax:

CALL LtsMenuG (Item\$(), Choice%)

Item\$() is a conventional string array containing a list of the menu items. The maximum length for any menu item is 78 characters.

. Where

Prompt\$0 is a paral Item\$0 is a convent

each menu item relat

Syntax:

a choice.

menu item. A choi The second line is us menu system. A lis

pressing Enter, or b

then Choice% will return a 0. Choice% returns the selection made by the user. If the user presses Esc,

Comment:

BASIC Routines

begin with a unique first letter. If you make this modification, though, be aware that each choice must to have it return as soon as the first letter of a choice has been pressed to the calling program. However, comments in the source code show how As shipped, LtsMenuG waits until the Enter key is pressed before returning

the first and last menu item respectively. LtsMenuG recognizes the Home and End keys and places the cursor on

returning to the c

As shipped, Lts21v

Comment:

then Choice% will

The example program DEMOLTS.BAS shows this routine in use.

See Also:

Lts2MenuG

See Also:

The example program the first and last menu Lts2MenuG recognize must begin with a uni pressed. If you make show how to have it r

LisMenuG

ON BRIMAIN HEADERS

SC survivine TSKENU BAS

Lts2MenuG

3ASIC subroutine contained in LTSMENU.BAS

6 J.S. Nokalie"

S and pressing Enter, oly on a single line.

The second line is used to provide an explanation of the currently selection menu item. A choice can be made either by using the arrow keys and pressing Enter, or by pressing a key that corresponds to the first letter of The Lts2MenuG routine provides a double-line Lotus 1-2-3 "look alike" menu system. A list of choices is displayed horizontally on a single line. a choice.

Syntax:

of the menu items.

CALL Lts2MenuG (Item\$(), Prompt\$(), Choice%)

e user presses Esc.

ed before returning

ce code show how tas been pressed

Item\$0 is a conventional string array containing a list of the menu items.

Prompt\$0 is a parallel string array containing the help or explanation of each menu item relating to the items in the Item\$() array. Choice% returns the selection made by the user. If the user presses Esc., then Choice% will return a 0.

BASIC Routines

Comment:

each choice must

ces the cursor on

ne in use.

As shipped, Lts2MenuG waits until the Enter key is pressed before returning to the calling program. However, comments in the source code show how to have it return as soon as the first letter of a choice has been pressed. If you make this modification, though, be aware that each choice must begin with a unique first letter.

Lts2MenuG recognizes the Home and End keys and places the cursor on the first and last menu item respectively.

The example program DEMOLTS.BAS shows this routine in use.

See Also:

LtsMenuG

MsgBoxG

BASIC subroutine contained in MSGBOX.BAS

programs that use Gr Mag Box G is shown

All of the culous for tradition are all initial values are all initial culous are all initi

■ Purpose:

a shadow effect to simulate depth. is always saved and it may be restored again later. The message box uses word wrap automatically centered on the screen. MsgBoxG provides a quick and attractive way to display a message, with The underlying screen

■ Syntax:

CALL MsgBoxG(Message\$, Wide%)

Where:

screen restored null, the most recently displayed message is cleared, and the underlying Message\$ is a single continuous string to be displayed. If Message\$ is

Wide% is the desired width of the text (up to 74).

Comments:

BASIC Routines

LOCATE should be used to set that before you call MsgBoxG. The top line of the MsgBoxG is placed at the current cursor line, so

screen, simply call MsgBoxG again with a null string then displays the message. To clear the message and restore the original message string. If it is not null, it first saves the underlying screen and When MsgBoxG is called, the first thing it does is check the length of the

be no way to restore the original screen. always be cleared before a new one is displayed. Otherwise, there will Be sure that you don't call MsgBoxG with a null string, unless it has already been called at least once before. Also be aware that the message should

additional columns are needed to accommodate the shadow text, and adds an extra blank space to make the text easier to read. Two The width is limited to 74 because MsgBoxG draws a border around the

CHARLEM MANAGE INC

BASIC Routines

The Graphics Workshop

SSC sectionine

The standards, with the manage by use

ed. If Messges is and the underlying

ent cursor line, so

estore the original

rdying screen and

k the length of the

order around the

e message should privise, there will

less it has already

ier to read. Two

BASIC Routines

All of the colors for this routine are defined in the GPDat%() array. Their values are all initialized in the file GETVIDEO.BAS. The benefit of isolating the color definitions to a single file is that you may customize them to your own preferences, and they will then be reflected in all of the programs that use GETVIDEO.BAS.

MsgBoxG is shown in context in the DEMOMENU.BAS example pro-

CRESCENT SOFTWARE, INC.

TSOFTHARE INC

4-35

Outline Text

a Purpose:

OudineText draws a

NightFall

BASIC subroutine contained in FADE.BAS

Purpose:

FadeEGA routine to accomplish the fading effect. the second video page to the current active video page. Images fade in from the top of the screen towards the bottom. This routine uses the NightFall simulates a gradual nightfall, by bringing random portions of

Where:

CALL OutlineText(X

corner of the first ch

The X% and Y% val

Syntax:

CALL NightFall

Comments:

of this routines' functioning. this routine, the effect is much more pleasing to the eye and doesn't appear rigid. See the demo program DEMOFADE BAS for a visual description when there is a large portion of the screen which will not be changed by background of the two screen images is going to be similar. It is advisable to use this routine and this effect when the overall Any time

Colr% is the color 0-

Angle% of 0 will dra Angle% is the angle Texts is the string of

BASIC Routines

■ See Also:

of great use in the posit length in pixels of the T A companion function, on using the Mult% an the routine to use integ

the original font defini To use the Mult % and Comments: Divide% is an in Mult% is an integer

GetOutlineWidth 8

SASC Subrouting

OutlineText

BASIC subroutine contained in OUTLTEXT.BAS

Purpose:

E rendom portions of This country and the in-

OutlineText draws a string using the Graphics Workshop Vector fonts.

Syntax:

CALL OutlineText(X%, Y%, Text\$, Angle%, Colr%, Mult%, Divide%)

■ Where:

The X% and Y% values represent the pixel coordinates for the upper-left corner of the first character to be drawn.

Text\$ is the string of the phrase to be drawn.

when the overall similar. Any time lot be changed by and doesn't appear a Visual description

Angle% is the angle in degrees at which the text will be drawn. Angle% of 0 will draw text in the normal text direction.

Colr% is the color 0-15 for the text to drawn.

Mult% is an integer multiplier for the size of the text.

BASIC Routines

Divide% is an integer divider for the size of the text.

Comments:

To use the Mult% and Divide% parameters to obtain a font size of 3/4 of the original font definition, set Mult % to 3 and Divide % to 4. This allows the routine to use integer arithmetic. See Chapter 6 for more information on using the Mult% and Divide% variables to obtain any size font. A companion function, GetOutlineWidth %, can be used to determine the length in pixels of the Text\$ before it is drawn to the screen. This can be of great use in the positioning of a text string.

See Alco.

GetOutlineWidth%

PositionBox

PCXCAP

BASIC program contained in PCXCAP.BAS

Purpose:

PCXCAP is TSR utility for capturing PCX images from just about any graphics mode program. To use PCXCAP, run PCXCAP.EXE from the command line, and the at a time when you are on a graphics screen, that you want to capture, press Alt-S and then type in an 8 letter filename.

Syntax:

CALL PositionBox(64

the XOR ability of th looks for cursor keys accepting user input t PositionBox is a comp

Where:

with variables of this structure are X1%, Vi defined in the include GWWindow is a BA

Style% is a boolean v otherwise a solid reg

Syntax:

PCXCAP

it a second time at the command line to remove it from memory. This is what you would type at the command line to start the utility. Type

Comments:

BASIC Routines

requires that you have another of our products, P.D.Q to recompile it. Although complete source code is provided for your amusement, it

only a portion of the image. image from the screen. Only in the EGA and VGA modes can you save SavePCXRegionVE to save only a portion of the screen. PCXCAP calls upon the SavePCX??? routines to actually save the PCX In this case PCXCAP calls upon

SpaceBar, and Esc.

Comments:

Eschessed & if the Esc The Esc key stops use values can be found in th The Enter key accepts t lower-right corner. The The SpaceBar toggles The Cursor keys move key is pressed. The keys accepted by

Position Box

BASIC subroutine contained in XORBOX.BAS

S from St. about any OXCAPEXE from the s graphics screen, that

d n Pr

n 8 letter fillename.

PositionBox is a complete routine for placing a box selector on screen and accepting user input to move and change the size of the box. The routine looks for cursor keys and selects a rectangular region on the screen using the XOR ability of the LineVE routine.

Syntax:

CALL PositionBox(GWWindow AS Window1, Style%, EscPressed%) IF ESCPressed% 0 THEN PRINT "Esc was pressed"

start the utility. Type

от тетоту.

Our amusement, it

to recompile it.

ually save the PCX nodes can you save XCAP calls upon

GWWindow is a BASIC TYPE structure. The structure Window1 is defined in the include file GWDECL.BAS. The variables in this record structure are X1%, Y1%, X2%, and Y2%. An example of programming with variables of this kind is shown at the beginning of this chapter.

Style% is a boolean variable. If it is 0, the outline of a box will be used; otherwise a solid region is used to display the area.

BASIC Routines

ESCPressed% is used to determine whether or not the Esc key was

Comments:

The keys accepted by this routine are any of the Cursor keys, Enter, SpaceBar, and Esc. The routine remains active until the Esc or Enter SpaceBar, and Esc. key is pressed.

The Cursor keys move the active corner.

The SpaceBar toggles which corner is active, either the upper-left or lower-right corner. The routine starts with the upper-left corner.

The Enter key accepts the box position as it is on the screen. The new values can be found in the GWWindow array.

A value of -1 is returned in The Esc key stops use of the routine. EscPressed% if the Esc key was pressed. 4 - 39

T SOFTWARE, DVC.

them to your own propression of the programs that use GI

PullDownG

BASIC subroutine contained in PULLDOWN.BAS

Purpose:

related groups of items. the underlying screen and accommodates a separating divider between being able to display more than one list of choices, it also always saves PullDownG is a complete graphics mode multiple-menu subprogram with many important capabilities including full support for a mouse. Besides

See Also:

PullDownG is explanted in Chapter I.
Menus in

PullDnMSG

whereby it is polled periodically to see if a choice has been selected Finally, PullDownG may be operated in a unique multi-tasking mode, Furthermore, selected menu items may be allowed or disallowed at will.

Syntax:

CALL PullDown6(Choice\$(), Status%(), Menu%, Choice%, Ky\$, Action%)

■ Where:

BASIC Routines

by a separating line and will not be selectable by the user. each menu. If any element contains a hyphen only ("-"), it will be replaced Choice\$0 is a two-dimensional array containing the list of choices for

the element that corresponds to a given item in the Choice\$() array. are active. Choices can be deactivated by assigning a non-zero value to Status %0 is a parallel, two-dimensional array that indicates which choices

may also be pre-loaded to force a given menu to be displayed initially. Menu% indicates which menu was active when a choice was selected, and

to force a given choice to be highlighted initially. Choice% indicates which choice was selected, and may also be pre-loaded

determine if the user pressed < Esc > to exit the menu system. Ky\$ holds the last key that was pressed by the user. This is used to

Menus" in Chapter 1. the Action% parameter are discussed in the section entitled "Multi-Tasking Action% tells PullDownG how it is being used. The possible values for

Comments:

All of the colors for this routine are defined in the GPDat%() array. Their values are all initialized in the file GETVIDEO.BAS. The benefit of isolating the color definitions to a single file is that you may customize The benefit of

CHECKET SHAWARE INC.

BASIC Routines

The Graphics Workshop

BASIC Subourne DOMNINGRS

assu submortan with for 8 mouse. Besides s, if also always saves ting divider between

See Also:

uses.

Menus" in Chapter 1, and two complete demonstrations are also provided. DEMOPULL.BAS shows the minimum setup required for calling Pull-DownG, and DEMOMENU.BAS illustrates some of its more advanced PullDownG is explained in depth in the section entitled "Multi-Tasking

PullDnMSG

or disallowed at will. multi-tasking mode,

3s been selected.

· Kyl. Actions

list of choices for), it will be replaced

cates which choices 1 non-zero value to also be pre-loaded

was selected, and

Dices array.

played initially.

This is used to

system.

ossible values for d "Multi-Tasking CRESCENT SOFTWARE, INC.

T SOFTWARE, INC.

may customize

R. () array. Their The benefit of

PullDnMSG

contained in PULLDNMS.BAS **BASIC** subroutine

Purpose:

underlined by the PullDnMSG routine when shown on screen. related groups of items. The PullDnMSG routine mimics as closely as the underlying screen and accommodates a separating divider between many important capabilities including full support for a mouse. Besides being able to display more than one list of choices, it also always saves PullDnMSG is a complete graphics mode multiple-menu subprogram with Each of the menu items has a hotkey assigned to it. The hotkey is then possible the functionality of the Microsoft Windows menuing system.

> that use GETVIDEO own preferences, an

There are two routine

All of the colors for I

Comments:

values are initialized

the color definitions i

Menus" in Chapter

the Action % paramet Actions, tells Pulls

whereby it is polled periodically to see if a choice has been selected. Finally, PullDnMSG may be operated in a unique multi-tasking mode, Furthermore, selected menu items may be allowed or disallowed at will

CALL PullDnMSG(Choice\$(), Status%(), Menu%, Choice%, Ky\$, Action%)

Where:

BASIC Routines

by a separating line and will not be selectable by the user. each menu. If any element contains a hyphen only ("-"), it will be replaced Choice\$0 is a two-dimensional array containing the list of choices for

■ See Also:

replace the main INI MenuKeyMSG routin Alt-key combinations har on-screen before user interface. The

so that your program

the users input. The

minimum menuing s

MenuKeyMSG routi

than the letter's position. in its respective Status %() element. Note that you store a value one less the 2nd letter is to be the hot letter, then you would store the value 1 * 256 position of the hotkey is stored in the high byte of the Status % () array. the element that corresponds to a given item in the Choice\$() array. The are active. Choices can be deactivated by assigning a non-zero value to Status %0 is a parallel, two-dimensional array that indicates which choices

may also be pre-loaded to force a given menu to be displayed initially. Menu% indicates which menu was active when a choice was selected, and

to force a given choice to be highlighted initially. Choice% indicates which choice was selected, and may also be pre-loaded

menu item currently selected can be determined. Then by looking at the Menu% and Choice% variables, the menu and example, if Ky\$ is equal to CHR\$(13) then the user has pressed Enter. determine if the user pressed Esc or Enter to exit the menu system. Ky\$ holds the last key that was pressed by the user. This is used to

■ 4 - 42

CRESCENT SOFTWARE, INC.

CHESTER MANSER

PUIDAMSBAS

the Action % parameter are discussed in the section entitled "Multi-Tasking Action% tells PullDnMSG how it is being used. The possible values for Menus" in Chapter 1.

Small representation with for a move. Besides है. में ब्रेफ़ बोफ्नाड saves ating divider baween mimics as closely as Ws menning system. . The hothey is then

All of the colors for this routine are defined in the GPDat%() array. Their values are initialized in the file GETVIDEO.BAS. The benefit of isolating the color definitions to a single file is that you may customize them to your own preferences, and they will then be reflected in all of the programs that use GETVIDEO.BAS.

user interface. The BarPrintMSG routine allows you to place the menu bar on-screen before the user has even entered the menu system. The There are two routines which work with PullDnMSG to provide a complete MenuKeyMSG routine interprets the user's keystrokes and filters out so that your program can interpret it. The MenuKeyMSG routine should replace the main INKEY\$ statement in your program that is waiting for the users input. The program example DEMOPLMS BAS shows a the MenuKeyMSG routine otherwise returns the keystroke that was pressed Alt-key combinations which can be used to start the menu system. minimum menuing setup for PullDnMSG.

> or disallowed at will. multi-tasking mode,

On Screen.

is been selected.

Kyl, Actions)

See Also:

list of choices for , it will be replaced

Status %() array. If e the value 1 * 256

e a value one less

oices() array. The

non-zero value to

was selected, and

played initially.

also he pre-loaded

This is used to enu system. For

s, the menu and

as pressed Enter.

BASIC Routines

PullDownG

RandomFade

contained in FADE.BAS **BASIC** subroutine

Purpose:

may be the same, it's SetOPFont is design

the font. This routing reparate information SetGPFont

Purpose:

RandomFade randomly fades in portions of the second video page to the currently visible video page. This routine uses the FadeEGA routine to accomplish the effect of fading in the image.

Syntax:

CALL RandomFade

Comments:

when there is a large portion of the screen which will not be changed by this routine, the effect is much more pleasing to the eye and doesn't appear rigid. See the demo program DEMOFADE.BAS for a visual description of this routine's functioning. background of the two screen images is going to be similar. It is advisable to use this routine and this effect when the overall Any time

Where:

DesiredFontNum%i

will be honored, prov

Syntax:

CALL SetGPFor

BASIC Routines

■ See Also:

in the "Standard | 0 just prior to loading

There is only one time

Comments:

CHEWENT WHEN THE INC.

S.S. Subouhe

SetGPFont

BASIC subroutine contained in DRAWTEXT.BAS

Purpose:

Fades GA routing to

SetGPFont is designed to change between loaded fonts. Each font has separate information about its width and height. Even though these values may be the same, it's always advisable to use the proper information for the font. This routine sets up all of this information and requires only one

■ Syntax:

CALL SetGPFont(DesiredFontNum%)

t when the overall similar. Any time I not be changed by eard doesn't appear a visual description

■ Where:

DesiredFontNum% is the font number you wish to change to. This value will be honored, provided that the number is within the range of available

Comments:

just prior to loading the font using the LoadFont routine. This is shown in the "Standard Code" section of Chapter 1. There is only one time when you are required to call this routine. That is

See Also:

SetGWFont

SetGPSpacing

contained in OUTLTEXT.BAS BASIC subroutine

I purpose:

SelOVFoot is design

SetGWFont

Purpose:

SetGPSpacing is used to set the spacing for the GraphPak fonts to be drawn with either DrawText or StepText.

Syntax:

CALL SetGPSpacing(SpacingH%, SpacingV%)

Where:

SpacingH% sets the spacing between characters. This value is measured in pixels for the base font size.

Where:

DesiredFontNum%

will be honored, prov

Syntax:

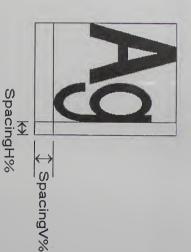
CALL SetSWFont(Des

information and requ the proper informat Even though these v fonts. Each font ha

Spacing V% sets the spacing between lines. This value starts at the position of the descender of a lower case letter such as 'g', and counts down in pixels.

Comments:

BASIC Routines



■ See Also:

in the "Standard Co just prior to loading

There is only one tim

Comment

SetGPFont

See Also:

SetGWSpacing

THEORY SHAWE DC

SetGWFont

BASIC subroutine contained in OUTLIEXT.BAS

Purpose:

fonts. Each font has separate information about its width and height. Even though these values may be the same, it's always advisable to use the proper information for the font. This routine sets up all of this SetGWFont is designed to change between loaded Graphics Workshop information and requires only one step.

■ Syntax:

is value is measured

CALL SetGWFont(DesiredFontNum%)

■ Where:

value starts at the as 'g', and counts

DesiredFontNum% is the font number you wish to change to. This value will be honored, provided that the number is within the range of available fonts.

Comment:

There is only one time when you are required to call this routine. That is just prior to loading the font using the LoadFont routine. This is shown in the "Standard Code" section of Chapter 1.

BASIC Routines

See Also:

SetGPFont

SetGWSpacing

BASIC subroutine contained in OUTLTEXT.BAS

Purpose:

SetVideo

Purpose:

SetGWSpacing is used to set the spacing for the Graphics Workshop fonts drawn by the OutlineText routine.

Syntax:

CALL SetGWSpacing(SpacingH%, SpacingV%)

Syntax:

CALL SetVideo

select the best possib SetVideo sets the scre

This routine replaces

Where:

in pixels for the base font size. SpacingH% sets the spacing between characters. This value is measured

position of the descender of a lower case letter such as 'g', and counts Spacing V% sets the spacing between lines. down in pixels. This value starts at the

screen mode to be us

The code in GETVII

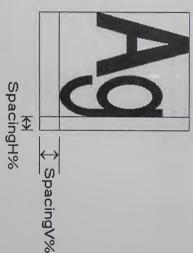
Add this routine as th Comments:

Servideo uses the B

you want to use any o Workshop has replace een duplicated by

Comments:

BASIC Routines



See Also:

SetGPSpacing

CHI SAWALSHO, LISTORED

ut ext drawn with the B streen mode. It sets ele It sets element 50 which

It sets element 49 which elements 43 and 44 whi which can be helpful v appropriate SCREEN St Not only does the SetVice in Chapter I will fulfill

up the GPDat%() array. element 31 of the GPD contains the screen mo SetVideo expects that

the BASIC graphius III VIEWPCX.BAS wh: Graphics Worksh, p. r. If you are designing

SetVideo

BASIC subroutine contained in SETVIDEO.BAS

Purpose:

SetVideo sets the screen mode and important screen information variables. This routine replaces the BASIC SCREEN statement as it allows you to select the best possible screen mode available with your monitor setup.

Syntax:

CALL SetVideo

Comments:

value starts at the as 'g', and counts

Add this routine as the last line of the standard code shown in Chapter 1. The code in GETVIDEO.BAS, or similar code which will determine the screen mode to be used, needs to be executed prior to calling SetVideo.

SetVideo uses the BASIC SCREEN statement even though Graphics you want to use any of the graphics routines like PAINT which have not been duplicated by routines in Graphics Workshop, you will need the BASIC SCREEN statement in order to use the BASIC PAINT statement. If you are designing a program that does not need any of the BASIC graphics statements, you can modify a version of SetVideo to use the Graphics Workshop routines. An example of this is the demo program VIEWPCX.BAS which creates a very small VIEWPCX.EXE by not using Workshop has replacement routines which set the same screen modes. the BASIC graphics libraries.

BASIC Routines

contains the screen mode desired. See Appendix C for a description of element 31 of the GPDat%() array. The code in GETVIDEO.BAS sets SetVideo expects that the GPDat%() array exists and that element 31 up the GPDat%() array. Note that including the standard code described in Chapter 1 will fulfill these requirements.

appropriate SCREEN statement, but it also sets other GPDat%() variables Not only does the SetVideo routine set the graphics screen mode using the It sets element 49 which holds the aspect ratio for the current screen mode. It sets element 50 which holds the number of colors available in the current screen mode. It sets element 71 which holds the current character height of text drawn with the BASIC PRINT statement or any of the GPrintOVE elements 43 and 44 which hold the current pixel resolution of the screen. which can be helpful when programming for graphics mode.

ShadeHorizon

bit pattern from one Shade Horizontal sha

ShadeH

contained in SHADEH.BAS **BASIC** subroutine

Purpose:

ShadeH shades a region of the screen with a gradually changing bit pattern from one color to another.

CALL ShadeH(ULCol, ULLine, LRCol, LRLine, NumColors%, StepChoice%,_ Colors%())

Where:

values range from 1 to 80, and line values range from 0 to 479 on a VGA These coordinates follow the mixed coordinate system where column monitor. region to shade. ULCol% and ULLine% specify the upper-left corner of the region to LRCol% and LRLine% specify the lower-right corner of the Together they specify the entire region to be shaded.

> NumColors% tells th with variables of this structure are X1%, Y GWWindow is a Br defined in the includ

NumColors% tells the routine how many color changes to go through.

BASIC Routines

are built almost randomly and use this value as the seed. StepChoice% tells the routine what the pattern will look like. The patterns

Colors%0 is an array of the colors for the routine

Comments:

and the Colors%(i) el

more solid as the rout

Colors%() is an

are built almost rand StepChoice% tells th

Comments;

solid, Colors %(1) will

the minimum value for the NumColors% parameter is 1. array should be one greater than the value of the parameter NumColors%; become the foreground color. The minimum dimension for the Colors %() solid, Colors %(1) will become the background color and Colors %(2) will more solid as the routine progresses. Once the color on screen becomes and the Colors %(1) element as the foreground color which will become This routine will start with Colors %(0) element as the background color,

ShadeHorizontal routine allows you to use any pixel locations is faster because it uses the mixed coordinate system. The routine ShadeH is a faster version of the ShadeHorizontal routine. However the

See Also: ShadeH, ShadeVertical

ShadeHorizonal routine

is faster because it use The routine ShadeH is a no minimum value for array should be one gre become the foreground

See Also:

ShadeHorizontal, ShadeV

CHESTER MANAGE INC.

ShadeHorizontal

BASIC subroutine contained in SHADEH.BAS

■ Purpose

ShadeHorizontal shades a region of the screen with a gradually changing bit pattern from one color to another.

■ Syntax:

CALL ShadeHorizontal(GWWindow AS Window1, NumColors%, StepChoice%,_
Colors%())

Where.

tight corner of the

Sion to be shaded, tem where column 0 to 479 on a VGA

GWWindow is a BASIC TYPE structure. The structure Window1 is defined in the include file GWDECL.BAS. The variables in this record structure are X1%, Y1%, X2%, and Y2%. An example of programming with variables of this kind is shown at the beginning of this chapter.

NumColors% tells the routine how many color changes to go through.

StepChoice% tells the routine what the pattern will look like. The patterns are built almost randomly and use this value as the seed.

BASIC Routines

Colors %0 is an array of the colors for the routine.

like. The patterns

es to go through.

Comments:

This routine will start with Colors %(0) element as the background color, and the Colors%(1) element as the foreground color which will become more solid as the routine progresses. Once the color on screen becomes solid, Colors%(1) will become the background color and Colors%(2) will become the foreground color. The minimum dimension for the Colors %() array should be one greater than the value of the parameter NumColors %; the minimum value for the NumColors % parameter is 1.

The routine ShadeH is a faster version of the ShadeHorizontal routine. It However the ShadeHorizontal routine allows you to use any pixel locations. is faster because it uses the mixed coordinate system.

See Also:

contal routine. It

HOWEVER THE

1 Colors % (2) will for the Colors % () er NumColors %;

ackground color, hich will become ShadeH, ShadeVertical

ShadeV

contained in SHADEV.BAS BASIC subroutine

Purpose:

Shade Vertical shades

pattern from one col

CALL ShadeVer

Purpose:

ShadeV shades a region of the screen with a gradually changing bit pattern from one color to another.

Syntax:

CALL ShadeV(ULCol, ULLine, LRCol, LRLine, NumColors%, StepChoice%, Colors%())

shade. values range from 1 to 80, and line values range from 0 to 479 on a VGA These coordinates follow the mixed coordinate system where column region to shade. ULCol% and ULLine% specify the upper-left corner of the region to LRCol% and LRLine% specify the lower-right corner of the Together they specify the entire region to be shaded.

NumColors% tells the routine how many color changes to go through.

NumColors% tells th

with variables of this structure are X1%, Y GWWindow is a Br

defined in the includ

StepChoice% tells th

BASIC Routines

StepChoice% tells the routine what the pattern will look like. The patterns are built almost randomly and use this value as the seed.

Colors %0 is an array of the colors for the routine

Comments:

array should be one greater than the value of the parameter NumColors%; the minimum value for the NumColors% parameter is 1. solid, Colors %(1) will become the background color and Colors %(2) will more solid as the routine progresses. Once the color on screen becomes This routine will start with Colors%(0) element as the background color, and the Colors%(1) element as the foreground color which will become become the foreground color. The minimum dimension for the Colors % ()

> and the Colors %(1) el more solid as the routi

his routine will start

Colors%() is an

Comments:

become the foreground

old, Colors %(1) will

faster because it uses the mixed coordinate system. However the Shade-Vertical routine allows you to use any pixel locations. The routine ShadeV is a faster version of the ShadeVertical routine. It is

See Also: Shade-Horizontal, Shade

Vertical routine allows y faster because it uses th The routine ShadeV is a the minimum value for array should be one gre

See Also:

ShadeH, ShadeVertical

ShadeVertical

BASIC subroutine contained in SHADEV.BAS

Purpose:

Shade Vertical shades a region of the screen with a gradually changing bit pattern from one color to another.

Syntax:

CALL ShadeVertical(GWWindow AS Windowl, NumColors%, StepChoice%,

let of the region to right corner of the

O to 479 on a VGA gion to be shaded, tem where column

An example of programming GWWindow is a BASIC TYPE structure. The structure Window1 is defined in the include file GWDECL.BAS. The variables in this record structure are X1%, Y1%, X2%, and Y2%. An example of programs with variables of this kind is shown at the beginning of this chapter.

NumColors% tells the routine how many color changes to go through.

StepChoice% tells the routine what the pattern will look like. The patterns are built almost randomly and use this value as the seed.

like. The patterns es to go through.

Colors%() is an array of the colors for the routine.

Comments:

This routine will start with Colors %(0) element as the background color, and the Colors %(1) element as the foreground color which will become Once the color on screen becomes become the foreground color. The minimum dimension for the Colors%() solid, Colors %(1) will become the background color and Colors %(2) will array should be one greater than the value of the parameter NumColors%; the minimum value for the NumColors% parameter is 1. more solid as the routine progresses.

The routine ShadeV is a faster version of the ShadeVertical routine. It is faster because it uses the mixed coordinate system. However the Shade-Vertical routine allows you to use any pixel locations.

See Also:

ShadeHorizontal, ShadeV

BASIC Routines

for the Colors %() er NumColors%; J Colors #(2) will

hich will become

sackground color.

cal routine. It is

wever the Shade-

CRESCENT SOFTWARE, INC.

4 - 53

Step lext

contained in DRAWTEXT.BAS **BASIC** subroutine

a Purpose:

a selection has been

unique multi-tasking saves the underlying many important capo Vent Menu G is a com VertivenuG

Purpose:

GraphPak Professional. StepText is used to draw a text string using the fonts available with

Syntax:

CALL StepText(X%, Y%, Text\$, Angle%, Colr%, SizeMultiplier#)

Where:

The variables X% and Y% are pixel positions of the upper-left corner of the first character of the string to be drawn.

Where:

Items\$0 is a conven

ly of menu choices.

Choice% indicates w

to force a given choic

Text\$ is a string of the phrase to be drawn.

placed right side up. of 0 draws text straight across the screen. All individual letters will be Angle% is the angle in degrees at which the phrase will step. An angle

activate the shadow effect. have shadows drawn underneath them. Adding 128 to the color will Colr% is the color of the text. Both DrawText and StepText phrases can

BASIC Routines

of the font. the original definition of the font, 2# = twice as big, .75 = 3/4 the size SizeMultiplier# is the size of the font. For example, 1# = same size as

Comments:

about the GPDat%() array. The color for the shadow effect mentioned above can be changed by modifying the variable GPDat%(14). See Appendix C for information

See Also:

GetTextWidth%, DrawText

ON THE MAN MATTER

physians that use GET

mem to your own prefer

olating the color defi

values are all initialize

All of the colors for this Comments: Chapter L. values are described in

Action% tells VenMen Kys holds the last key be on line twenty. I

is, if BoxBot% is set t

BoxBot% is the button

the menu width

MaxLen% is th

VertMenuG

BASIC subprogram contained in VERTMENU.BAS

Purpose:

The activity with

VertMenuG is a comprehensive menu subprogram for graphics mode with many important capabilities including full support for a mouse. It always saves the underlying screen. Further, VertMenuG may be operated in a unique multi-tasking mode whereby it may be polled periodically to see if a selection has been made.

■ Syntax:

upper-left corner of

■ Where:

vill step. An angle dual letters will be

epText phrases can

75 = 3/4 the size

Items\$0 is a conventional (not fixed-length) string array containing the list of menu choices. Choice% indicates which choice was selected, and may also be pre-loaded to force a given choice to be highlighted when the menu system is accessed

MaxLen% is the maximum length of any menu choice, thus establishing the menu width. Choices that are longer than MaxLen% will be displayed truncated.

BoxBot% is the bottom screen line that the window is to extend to. That is, if BoxBot% is set to twenty, then the bottom border of the menu will be on line twenty. Notice that the upper-left corner of the menu is established by the current cursor location.

Ky\$ holds the last key that was pressed by the user.

for information

Action% tells VertMenuG how it is being used. The different possible values are described in the section entitled "Multi-Tasking Menus" in Chapter 1.

Comments:

All of the colors for this routine are defined in the GPDat%() array. Their values are all initialized in the file GETVIDEO.BAS. The benefit of isolating the color definitions to a single file is that you may customize them to your own preferences, and they will then be reflected in all of the programs that use GETVIDEO.BAS.

BASIC Routines

WhichPCXScr

Purpose:
WhichPCXScreen in
mode that should be

Where:

Arrays contains the

by calling the routing

WhichScreen% retu value of WhichScree which is explained in

Comments:

This routine evaluates screen mode from the scaleblased in this coun mode, it is done as An example of using example program, and

B Syntax:

CALL WhichPCXScn

VertMenuG is explained in depth in the section entitled "Multi-Tasking Menus" in Chapter I, and a complete demonstration is provided in the DEMOVERT.BAS example program. This routine is also used by the DEMOMENU.BAS example program.

BASIC Routines

■ 4-56

CRESCENT SOFTWARE, INC.

THE WALL WASHINGTON

A Partie of the Partie of the

WhichPCXScreen

BASIC subroutine contained in PCXHEADR.BAS

Purpose:

WhichPCXScreen interprets a PCX file header and determines the screen mode that should be used.

■ Syntax:

CALL WhichPCXScreen(Array\$, WhichScreen%)

■ Where:

Array\$ contains the header information for the .PCX file. Array\$ is set by calling the routine OpenPCXFile%.

WhichScreen% returns the suggested screen mode for the .PCX file. The value of WhichScreen% follows the values of the GPDat%(31) variable which is explained in Appendix C.

Comments:

screen mode from this information. Even though not all of the information calculated in this routine is not used by the routine to determine the screen This routine evaluates information in the header and determines the desired mode, it is done to give a complete breakdown of the PCX file header.

BASIC Routines

An example of using this routine is contained in the VIEWPCX.BAS example program, and the BASIC routine DisplayPCXFile.

OTHER FILES ON THE DISK

GWDECL.BAS FADEDATA.GW BC7END.GW BC7START.GW GWFONT.GW GPFONT.GW SALES2.PCX SALESI.PCX EXTERNAL.GW COMMON.GW MOUSE.PCX MOUNTAIN.PCX CGA.PCX ZOOMED.PCX XORBOX.PCX FONTS3.PCX FONTS2.PCX FONTS1.PCX FONTEDIT. PCX DEMOSHAD.PCX GW7.QLB GW7.LIB GW.QLB GW.LIB **QSEGUE.IN** HELV.QFN OLDENG.GPK FUTURE.GPK TROM12.GPK HELV8.GPK HELV12.GPK FUTURE.GFN TROM12.GFN OLDENG.GFN HELV8.GFN HELV12.GFN FILENAME EGA .PCX file for Demonstration List of EXTRN's for Graphics Workshop Initialization for GW Vector Fonts Assembler Include file for BC7 Compatibility Assembler Include file for BC7 Compatibility File of COMMON's for Graphics Workshop EGA .PCX file for Demonstration EGA .PCX file for Demonstration CGA .PCX file for Demonstration Initialization for GraphPak Fonts Data File for the FadeEGA Routine DECLARES for Graphics Workshop HERC .PCX file for Demonstration EGA .PCX file for Demonstration Graphics Workshop Library for BC7 Graphics Workshop Quick Library for BC7 Graphics Workshop Quick Library Graphics Workshop Library QuickSegue Input File 64 pt Helvetica GW Vector Font GraphPak Font Definition 8 pt Helvetica GraphPak Font 12 pt Future GraphPak Font 12 pt Times Roman GraphPak Font 12 pt Old English GraphPak Font 12 pt Helvetica GraphPak Font DESCRIPTION

BASIC Routines

QuickSegue Chapter 5: QuickSegue July Party for BC7 for

QuickSegue ON THIMMELLANDED Purpose: Where: for execution to add to a Purpose: CLEAR buffer Details About The Sc buffer is the number of Clears a graphics memo About The Script Li DO dotype The statements in the comment lines in the A script used by Qui
A script used by Qui
can create an ASCII t
can create an enu in
on the Files menu in
the script file it canno
the script file it canno Option is a property of the analysis of the an QuickSegue is a programmable slide show program which combines transfer of .PCX files with simple graphics routines. QuickSegue accepts You have the ability to load .PCX files and bring them forward to the screen in many different fashions. You can also annotate your .PCX files with text. QuickSegue also gives you the ability to add your own routines a script input file in a special format described below. QuickSegue will read the script file and manipulate the screen according to its statements. to be executed at an appropriate time within the QuickSegue script.

About The Script Language

A script used by QuickSegue file can be any plain ASCII text file. You can create an ASCII text file by loading a document using the Load option on the Files menu in QuickBASIC. QuickSegue will ignore any lines in the script file it cannot understand, so you may place blank lines, or even comment lines in the script file.

The statements in the QuickSegue script language are:

CLEAR

INTERLUDE

LOCATE SEGUE

SEND PAUSE

PRINT

Details About The Script Language

CLEAR buffer

Purpose:

Clears a graphics memory buffer which has been previously loaded.

Where:

buffer is the number of buffers to clear.

Purpose:

Allows programmers to add their own sections of code to QSEGUE.BAS for execution to add to a presentation.

QuickSegue

Where:

dotype is a number which is used to select one of the pre-programmed sections to execute. Currently only dotype = 1 is defined, and it moves an exclusive-OR box across the screen.

Where:

subtype specities the seguetype specifies the

have subtypes.

purpose:

Transfers a graphics

screen in one of man

INTERLUDE number "anystring"

Purpose:

Starts one of the interlude types. Two pre-written interludes exist.

Director's Clicker, using the phrase specified by anystring. number is the number of the interlude. Number = 1 brings up a Movie

LOAD "filename" buffer

Purpose:

Loads a .PCX graphics file into a memory buffer

Where:

is specified, QSEGUE.BAS will assume the current directory. filename is any valid filename including paths and extensions. If no path

> Purpose: SEND buffer

machine using the QS speed of the transitio delay specifies a time solid colored screen in color specifies the col

graphics file into. buffer is the number of one of the available buffers for loading the .PCX

LOCATE horizontal vertical

Where:

bring the PCX image

screen. Nothin

executes. It shou Sends a compres

graphics file to be displ buffer is the number

Purpose:

QuickSegue

command the size of the text string to be drawn is unknown. uses words to describe the position of a string since at the time of the locate Locates an internal cursor for the positioning of text strings. This routine

horizontal is an X axis positioning command. A positioning statements are "Left", "Middle", "Right" Available horizontal

vertical is a Y axis positioning command. statements are "Top", "Middle", "Bottom "Bottom" Available vertical positioning

Where:

Prints a text string at the

Purpose:

PRINT "anystring" on h

Purpose: PAUSE

Waits for the user to pre

CHECKETT SHAWARE, INC.

other is the color for dra anystring is any string e

SEGUE seguetype subtype color delay

and and a motor

Purpose:

Transfers a graphics screen from the background screen to the visible screen in one of many geometric fashions.

merludes exist.

seguetype specifies the type of geometric transition to be made.

subtype specifies the subtype of the above seguetype. Not all segue types have subtypes.

1 brings up a Movie

string

color specifies the color to paint with for those subtypes which bring in a solid colored screen instead of a .PCX image from the background screen.

delay specifies a time delay which can be used to uniformly control the speed of the transitions. The time is independent of the speed of the speed of the transitions. The time is independent of the speed of the machine using the QSEGUE program and is measured in milliseconds.

SEND buffer

Purpose:

ensions. If no path

r loading the .PCX

Nothing is physically seen by the user when this command. It should be followed by a SEGUE statement in the script to Sends a compressed graphics file from a buffer to the background video bring the PCX image to the screen. executes.

Where:

buffer is the number of one of the buffers which contains the .PCX graphics file to be displayed.

PAUSE

Purpose:

Waits for the user to press a keystroke.

PRINT "anystring" color

Purpose:

Prints a text string at the specified location in a helvetica font.

Where:

anystring is any string enclosed by quotes.

color is the color for drawing the text string and is a number between 0 and 15.

CRESCENT SOFTWARE, INC.

QuickSegue

ngs. This routine time of the locate

lable horizontal

rical positioning

■ 5-3

Segue Types

using pictures to better explain the process by which the segue takes place. All of the segue types described here are also described in Chapter 4,

Segue 1 corresponds to a quick full screen transfer of the .PCX file.

Segue 2 corresponds to a inwardly imploding box transfer of the .PCX

replacement. with parts of the new .PCX file. The subtypes define the pattern of the Segue 3 corresponds to a random square replacement of the old screen

size of the blocks used to make up the line. lower-right corner using parts of the .PCX file. The subtypes define the Segue 4 corresponds to a diagonal fade from the upper-left corner to the

replaced at a time. with the new .PCX file. The subtypes define the number of lines being Segue 5 corresponds to a horizontal line replacement of the old screen

Segue 6 corresponds to a outwardly exploding box transfer of the .PCX

Vector Fonts

Chapter 6:

September 1

in the pattern of the

per-left corner to the subtypes define the

nt of the old screen inber of lines being

ansfer of the . PCX

Vanity of the PCX

fite. PCX file.

Vector Fonts

ST SOFTWARE IN

Vector Fonts OBSERVATION WAS THE command does not be prematurely and does not be selection is importunity to the vector and indicatelying image with text, and the mage, the text will be is any already existing in the above PAINT st both the line and the Because the line cross the border of the box the box. Now consid Because uncertains when drawing vector when drawing vector when drawing to underreath where the cample in the Quickle Because the Vector A Vector Foot is a se foot. Each letter co foot. Each letter co foot is drawn vector Foot is drawn filed in with the app SCREEN 9

LINE (75, 150) - (1

LINE (75, 75) - (11

PAINT (80, 80), 2,

A Vector Font is a series of line segments which make up the outline of a font. Each letter can have its own number of line segments. When a Vector Font is drawn, its outline is generated first. The outline font is then filled in with the appropriate color.

Because the Vector Font is dependent upon the PAINT command, it is necessary to understand a little about how the PAINT command works, and the problems which can occur while using it. A problem can occur when drawing vector fonts on a screen which already has graphics images underneath where the new text string will be drawn. Try the following example in the QuickBASIC editor:

SCREEN 9 LINE (75, 150) - (150, 75), 1 LINE (75, 75) - (125, 125), 1, B PAINT (80, 80), 2, 1

'turn on the graphics screen 'assume previously on the screen 'draw box to be painted 'paint the box

Because the line crosses through the box and the line is the same color as the border of the box, the PAINT statement is unable to completely fill the box. Now consider that the box is the outline of a font, and the line is any already existing graphic image on the screen. Note: The first color in the above PAINT statement is irrelevant. The problem occurs because command does not know that the two are separate and therefore stops prematurely and does not paint the entire box. For this reason, color selection is important when drawing a Vector Font. If you are going to be in the underlying image. This is good for two reasons. One, it will insure that your text is drawn completely. Two, when you annotate a graphic using the vector fonts over already existing images, use a color not used image with text, and the text uses a different color than parts of your image, the text will be easily discernable from the background image. both the line and the box are drawn in the same color.

Using the Vector Font Editor

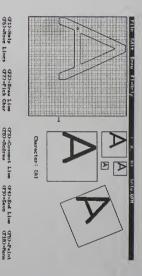
how the character will look at an angle. The Vector Font Editor is called FONT64.EXE and is a program for editing a font definition. The grid used in FONT64.EXE gives fonts up to a 64 x 64 point resolution. The Vector Font editor shows what the font will look like for 4 different sizes. One view of the font definition shows

Break Segment

Smulate Letter

Delete Segment Move Paint

Move Lines



Start the Vector Font Editor by typing:

Detailed Function L

다= 남 Redraw Letter

Set Baseline Paint Draw End Line Draw Connect Ends

Menu Items

New Font Items on the Files

This option clears the

FONT64

these is the <F10> key which is used to activate the menu system. The second is the <F1> key which is used to activate FONT64's help system. Once the Vector Font Editor has started, there are a couple of basic commands which can be used to create and edit vector fonts. The first of

Using the Menu System

available, selecting on any of the menu items at the top of the screen will activate the menu system. This section describes each of the options on each of the menus in the Vector Font Editor. The menu system is activated by pressing the <F10> key. If a mouse is

Vector Fonts

DOS Shell Load Simulate Font... Save Font. Save Font As.. Open Font.. New Font

■ 6-2

CRESCENT SOFTWARE, INC.

CHARLANG MARKET Save Font As... Save Font. greate the fort file. Al with the New Ford comm This option is used to go so further prompt will b

Open Font... it any changes have no

user will be prompted for executed then this funct specified by a previous This option is used to s so this need not be type of the four file. All year made since the last save The user is prompted i This option is used to lo

The Graphics Workshop

A POST OF THE PERSON OF THE PE ENE S SING September Stores stoks wathefor

<F7>> Delete Segment **Break Segment** Pick Character Simulate Letter Move Lines Move Paint

	DIAW	
	Line Draw	<f2></f2>
_	Connect Ends	<f3></f3>
_	End Line Draw	<f4></f4>
_	Paint Draw	<f5></f5>
_	Set Baseline	
	Redraw Letter	<f8></f8>

F1 = Help

Detailed Function Description Of Menu Items

Items on the Files menu

New Font

onts. The first of

couple of basic enu system. The 64's help system.

This option clears the memory of all letters of a font. The user is prompted if any changes have not yet been saved.

Open Font...

The user is prompted to save the current file if any changes have been made since the last save. The user will also be prompted for a filename This option is used to load a predefined font file for the purpose of editing. of the font file. All vector font files have the .QFN extension, by default, so this need not be typed.

Save Font...

ey. If a mouse is of the screen will of the options on This option is used to save the current font information to the filename specified by a previous open command. If no previous open has been executed then this functions the same as the Save As command and the user will be prompted for the filename to save the font under. Otherwise no further prompt will be given to the user before saving the font file.

Vector Fonts

Save Font As...

This option is used to give a font a new name, or to name a font created with the New Font command. The user is prompted for the name to be given to the font file. All vector font files have the .QFN extension.

Load Simulate Font...

use the Helv12.GFN font file. available GraphPak font files is in chapter 4. FONT64 will default and from GraphPak. GraphPak fonts have the .GFN extension, and a list of This option is used to load a simulation font from one of the fonts provided

Items on the Draw

two line segments, p

of the two line segme the first segment. N

Line Draw

is when either the End segment begins at the This option begins

is issued.

DOS Shell

This option is used to shell to DOS to utilize some of its functions

be prompted to save it. This option exits the font editor. If your file has not been saved, you will

Draw Another

This option replaces t

a Line Draw or Draw This option will draw

position. The routine

ammands to draw a

Items on the Edit menu

Pick Character

This option is used to determine which character's font description to edit.

Move Lines

the move lines and the lines will change on screen. Pressing < Enter > will finalize Lines, moving the cursor will also move the connecting point of these two cursor over the connecting point of these two lines. After selecting Move segments are connected at one point and the user must place the square This option is used to move the endpoint of two connected lines. Two line

Connect Ends

cursor position is whe

Move Paint

user must have the cursor over one of the paint points before using this command. Pressing < Enter > will finalize the move. This option is used to move the center point for the painting region.

way to the line segr

" Dureumon will

now boston original Line Draw This option draws a l

Delete Segment

be prompted to verify the removal of that segment. pressing the < SpaceBar > until that segment is the one with the different which make up the font definition. You select which segment to delete by This option will start by changing the color of one of the line segments This option will remove one of the line segments from the font description. Once the proper segment is selected, press < Enter >. You will

Paint Draw

which is not connected To the Draw menu. The I This option ends line dr End Line Draw

This option places a par command is to begin to

Break Segment

Vector Fonts

segment to break into two segments by pressing the <SpaceBar> until that segment is the one with the different color. Once the proper segment specified position. This option will start by changing the color of one of is selected, press < Enter >. At this point the line segment will be broken the line segments which makes up the font definition. into two and the cursor positioned halfway between the two endpoints of This option is used to insert line segments into the font structure at the You select which

Set Baseline

solution is to place more displayed at smaller size

section resulting in an in be placed in a character.

ON SAMUASO MADARD

letter such as 8 which a with and without descent vencal position. The ha This option sets the bas the first segment. Now the cursor keys can be used to position the midpoint To finalize the position of the midpoint for the of the two line segments. To finalize the postwo line segments, press the <Enter> key.

The Wilder and Secretary and a second

Items on the Draw Menu

Line Draw

Den savel, wa will

The line segment begins at the current cursor position and ends where the cursor is when either the End Line Draw command or the Connect Ends command This option begins the process of drawing a line segment.

Draw Another

This option will draw a line segment from the last cursor position where This option replaces the Line Draw option when a line draw is in progress. a Line Draw or Draw Another command was issued to the current cursor The routine is also prepared to receive additional Draw Another commands to draw a line from the current cursor position to wherever the cursor position is when those commands are issued. position.

Connect Ends

ted lines. Two line st place the square point of these two

er selecting Move

This option draws a line between the cursor position that existed when the original Line Draw (not Draw Another) command was issued to the current The next Line Draw command will create a line segment which is not connected in any cursor position. At the same time, this option ends the line drawing mode. It will restore the Line Draw option to the Draw menu. way to the line segment just drawn.

End Line Draw

ting region. The before using this This option ends line drawing mode. It will restore the Line Draw option to the Draw menu. The next Line Draw command will create a line segment which is not connected in any way to the line segment just drawn.

Paint Draw

he line segments with the different ter > You will

font description.

This option places a paint point marker at the position where the PAINT command is to begin to fill the outlined object. Multiple paint points can be placed in a character. A section of a font may be closed off from another section resulting in an incomplete painting of the entire character. When solution is to place more than one paint point in a letter's font definition. displayed at smaller sizes, this can happen near curves and edges.

Set BaseLine

structure at the

(u select which paceBar > until

This option sets the baseline for the font to be at the cursor's current vertical position. The baseline is the position which is used to align letters with and without descenders. A descender is the portion of a lower case This baseline etter such as 'g' which extends below most other letters.

Vector Fonts

NO ELECTO VINES OF

description. the height of a font, see Interlude1 and its use of the 2nd byte in the font can be used to determine the height of a vector font letter. To determine

Redraw Letter

this happens, using this command will clean up the screen. on screen. If a font definition is not made of completely enclosed objects, This option redraws the font definition for the currently selected character the PAINT command could paint outside the boundaries of the region. If

Items on the Help Menu

activate help. one method of obtaining help. The Help Menu has no items. Simply press < Enter > on this menu to There are no help options; there is only

Using Vector Fonts With Your Program

obtaining an integer numerator and denominator for some real number multiplies and divides which will be much faster than using floating point contained in A! is: arithmetic. will be multiplied by the font definition. which correspond to the numerator and the divisor of some fraction which variables to use for sizing the font. accepts a physical screen location to draw the string of text and two The routine that draws a vector font is called OutlineText. Almost any size font can be generated. These variables are integer values Basic can generate integer An algorithm for This routine

Numerator% = A! * 1000 Denominator% = 1000

provides a wide range of font sizes. The algorithm for drawing a string works just as fast with the Denominator equal to 1000 as when it is equal to 1. Using a Denominator of 1000

Appendices Appendices i ce cued character i en sed object, rice i the region. If othors; there is only Pext. This coutine 8 of text and two are integer values integer values generate integer sing floating point. An algorithm for onne real number the Denominator minator of 1000 TSOFTWARE DK

Appendices JAN SAWASIA MASSAL The fourth type is the nur used in must of the EGA The third byte is the com-the value 1 to represent PC PaintBrush IV uses v Appendix A: A PC Header portion The second byte is the file format it is. The this is whether or not palette. The version followed to the control of the contr used to describe the i following table: the fact that the fr PC Paintbrush fi. defines the file as a PC The first byte of the To start, like many do we call or bytes in length. Note: bytes in le pc Pallion read a ...
program can read a ...
the ... PCX format was
until this change, as
until this change, as
until this change, as Appelous time now, the for some time now, the for some time now, the format. The PCX of the pc paintbrush and a can read a ...

Appendix A: A PCX PRIMER

For some time now, the three letters PCX have come to mean bit mapped graphics, and if you want to display bit mapped graphics you use the PCX format. The PCX graphics format was created by ZSoft, the makers of PC Paintbrush and Publishers PaintBrush. Just about every graphics program can read a -PCX file and display it in some fashion. Until 1989, the -PCX format was left unchanged. We'll cover the -PCX format up designed for the 256 color mode of the VGA.

To start, like many data files, the .PCX file has a header portion which is used to describe the image. We will need to examine this header before we can do anything with the image. The size of the header is always 128 bytes in length. Note: If the header had been made variable in length there would have been no need to modify the standard to accommodate the 256 color VGA modes.

Header portion

The first byte of the header always contains the value 10 (0A hex) which defines the file as a PC Paintbrush PCX file. Use this in combination with the fact that the file has the .PCX extension to confirm that the file is a PC Paintbrush file.

The second byte is the version number. It tells you which version of the The version numbers which ZSoft has published are in the file format it is. The important information which can be derived from this is whether or not the file contains information about a re-mapped following table:

ON DESCRIPTION OF FORMAT	No Palette Information	Contains Palette	No Palette Information	Contains Delette
VERSION	0	2	m_	2

PC PaintBrush IV uses version 5 of the .PCX format.

The third byte is the compression scheme used. This byte will always have the value 1 to represent "run-length encoding", which is discussed later.

The fourth byte is the number of bits per pixel. This ignores color planes used in most of the EGA and VGA modes. Most of the time this byte will

and the VGA 256-color mode where this value is 8. The exceptions are the CGA 4-color mode where this value is 2.

comes in. Put simpl store that same infor

takes only two bytes to say the next 40 by

same, and the other

provides a compress

The fifth through twelfth bytes store the picture dimensions as 4 integer XMaximum, and YMaximum. The dimensions are shown in the order XMinimum, YMinimum,

and 350. was created on an EGA display, for example, these values would be 640 resolution possible for the machine on which the image was created. If it The next four bytes (13 through 16) store two integer values of the physical

> The next problem is I usually compress tha

palette is discussed in Appendix B. The next 48 bytes (17 through 64) store the palette information. The

Byte 65 in the file is reserved by ZSoft for future use

of color planes is set to 4; otherwise it is set to 1. When using one of the 16 color modes on the EGA or VGA, the number Byte 66 in the file tells you the number of color planes used by the image.

and the resulting value If the leftmost two bits data is to look at the

to 255 for counts. T decided to use the nu

there are no special byte. Since a data by

later. The simple wa

incorrect picture dimension, the value in Byte 67 was still correct, 80 bytes. 75 by 75 pixels in dimension. However, even in the .PCX files with that dimensions. One common mistake describes EGA screens as being only written incorrect values in the header bytes which describe the picture always been recorded correctly. information (ignoring multi-plane aspect). Byte 67 is very important. It defines the number of bytes per line the image A VGA screen which is 640 pixels across uses 80 bytes to store the Some versions of paint programs have The value in this byte has

The remaining 61 bytes in the header are unused

Data Portion

each video mode has a different way of storing pixels on the screen, and monitor used, and then interpret the data portion accordingly. is necessary to use the header information given above to determine the from, you cannot directly interpret the information in the data portion. closely resembles the memory of the video screen the image was taken the PCX file format stores information for each screen in a fashion that The data portion contains compressed bytes which hold pixel values. Since

devices to hold more information. screen has 80 bytes per scan line, 350 scan lines per video plane, and four For those unfamiliar with compression, it was invented to allow storage An EGA 16-color high resolution

ON BRIGHING MANAGED

LIPS. MA WALL LOOK AT THE TO SEE WAY & GUAY AND P. LAWAR SEE OF for end of line and end o raises the time it takes to This all-out compression if the first 3 bytes of the zeros, you would expect

zeros, what you actually

and even across a scan li planes. Some programs the blue plane's data is

a data value of 192 uses the remaining questions

than the image original

as two bytes inste in the cases when than 192, then we wo Table from 0 to 255. byte is going to be re

raphics, but it's rare

this compression conti

■ A-2

store that same information in less space. That's where data compression comes in. Put simply, if 40 bytes in a row are all the same, it is simpler to say the next 40 bytes are all the same and this is what the byte is. That takes only two bytes to represent: one to say that the next 40 bytes are the same, and the other to say what that byte is. Two bytes instead of 40 provides a compression of 95%. Unfortunately, the entire file doesn't It would be nice to usually compress that well; a typical average is around 60%. That's 91,200 bytes of information. video planes.

> Talles of the physical Se we created. If it Values would be 640

information. The

The next problem is how to represent a count and how to represent a data The people at ZSoft decided to use the numbers from 0 to 191 for data and the numbers 193 to 255 for counts. The value of 192 is not used, for reasons described The simple way of looking at a byte to see if it is a count and not byte. Since a data byte can have any of the possible values from 0 to 255. there are no special codes left to use for counts. data is to look at the leftmost two bits of the byte.

used by the image. VGA, the number

Der line the image O bytes to store the

in this byte has

nt programs have ens as being only CX files with that correct, 80 bytes.

If the leftmost two bits are set, then subtract the value 192 from that byte and the resulting value will be the count of how many times the following byte is going to be repeated. Note: this byte which follows can be any value from 0 to 255. If we had only a single byte with a data value larger In the cases where a single byte exists and it is larger than 192, it is stored as two bytes instead of one. This can be a problem in more complicated graphics, but it's rare for this compression scheme to use more memory than 192, then we would have to say here comes 1 byte and its value is X. than the image originally took up.

This compression continues for the remainder of the file. Now let's answer the remaining questions: How do I handle EGA video planes? Why isn't a data value of 192 used?

For each line, the blue plane's data is stored, followed by the green, red and intensity Some programs will compress across the boundaries of a plane and even across a scan line. For example, if the last 5 bytes of a line are zeros, you would expect to get a repeat count of 5 and then a zero. But, if the first 3 bytes of the next line or even the next video plane are also This all-out compression makes a compressed file smaller as a rule, but raises the time it takes to display a . PCX file because of the extra checking zeros, what you actually get is a repeat count of 8 followed by a zero. The EGA video planes are stored on a line-by-line basis. for end of line and end of plane.

data portion. It

to allow storage Mary.

mage was taken to determine the

xel values. Since 1 the screen, and in a fashion that To see why a data value of 192 isn't used, remember the .PCX format rules: We must look at the two leftmost bits to see if they are both ones.

Appendices

APPENDIX B: THE

scene, with a painti

A painter often uses

two colors for the sa colors in the same pl these colors on the (

make light blue with

selection of colors we

If they are, then subtract 192 from that byte to obtain a count. Both leftmost bits of 192 are ones. Since 192 - 192 is zero, and you can't have a count of zero, the value of 192 becomes meaningless and is not used in the .PCX format.

Appendices

A - 4

CRESCENT SOFTWARE, INC.

ON SPRINTING HARBERT

and By his one of four

is represented in the trill by the lower-case letter,

represented by the capit high-intensity value. In How The EGA Stores

in how these colors can

How The Palette Wo

with the right colors

and of the demo. to create a better we can have 4 shades go into more on how th the colors we won't no just eliminated the ne won't need a yellow, I

standard palette. Luci from 64 different colo

In particular, for a wa screens there is a pale On EGA and VGA sc

The EGA and VGA mo way they are completed y

and 2 bits for the Blu

time. Exchort here 16 a for the colors. To repo

The EGA has a maximus

APPENDIX B

APPENDIX B: THE PALETTE

seene, you can be sure there will be some blue, some green and some white. With a painting using oil paints, the artist has the ability to blend A painter often uses few colors to paint his masterpiece. If it is a water His whitecaps on the water can blend and In simplest terms he has two colors in the same place at the same time. A computer screen can't have A palette is a selection of colors we can choose from to paint our picture. two colors for the same pixel, but we do have a palette. make light blue with the water beneath. these colors on the canvas.

On EGA and VGA screens there are 16 color graphics modes. On these screens there is a palette of 16 colors. You don't always need every color. In particular, for a water scene you need blue, dark green and white. You won't need a yellow, red, purple, violet, bright green or a pink. You've ust eliminated the need for about half of the colors given to us in the standard palette. Luckily, the palette gives us the ability to replace any of the colors we won't need with another color. On the EGA you can select go into more on how these colors are structured later, but for now let's say we can have 4 shades of blue, 4 shades of green, and 4 shades of white. That's twelve colors to paint with instead of three. That's obviously going to create a better picture than if only three colors were used. Look at the end of the demo program DEMOPAL.BAS; it shows the effect of painting from 64 different colors; on the VGA you can select from 256,000. with the right colors.

How The Palette Works

The EGA and VGA monitors are the same in many ways. However, one way they are completely different is in the number of colors available and in how these colors can be accessed.

How The EGA Stores Palettes

The EGA has a maximum of 16 colors which can be displayed at any one time. Each of these 16 colors can be among one of the 64 possible choices for the colors. To represent 64 colors requires a 6-bit binary number. Among these 6 bits are 2 bits for the Red value, 2 bits for the Green value, represented by the capital letter, and the low-intensity bit is represented by the lower-case letter. Each of the values for the RGB value stored here is represented in the following fashion: 00RGBrgb. Each pair: Rr, Gg, and Bb, has one of four values: 00, 01, 10, or 11. This means that if the One bit for each color is used as high-intensity value. In the explanation below, the high-intensity bit i 2 bits for the Blue value.

awered in the demor image being displayed

number of bits which are set to a value of 1. The color 56 has only three bits which are 1's, while 55 has five bits which are 1's, thus the color 55 original color. to 1, you are selecting the intensity of the color you are adding to the will be considerably brighter than the color 56. generated. The same is true of the other colors. If you set any of its bits bits represented by Rr are both zero, then there is no red in the color The brightness of a color is determined by the overall

so it is not capable of telling you what are the values of the palette registers SetPaletteEGA will do this for you. The EGA uses "write-only" registers, number and a value to a routine which can set the palette. The routine Setting the EGA's palette is accomplished by sending a palette register currently in use. You will need to maintain this information for yourself

How The VGA Stores Palettes

the image appear to be Each of the colors fol

is pussible with just or These types of effects

displaying motion wit The Logo contained ii

to spend valuable C

color of individual pli these complex image you want to move it.

over them and it wou game. The terrain in One could also use P cadraw it to the scree drawing video data to native palette which i

the process by expanding the number of available colors. You can specify any of 64 values for each of the main colors red, green, and blue. This Some day computers will be giving us more colors to choose from, but values, the brighter the color will be. The VGA 256-color mode allows respectively. register and then three values for the Red, Green and Blue color values, provides an overall color palette of 256,000 colors. Graphics Workshop The VGA can store palettes the same way, but it has the ability to improve maximum which QuickBASIC can handle. for now that's the maximum we can accommodate on a PC and that's the any of its 256 colors to have one of the 256,000 colors mentioned above. has a routine called SetPalTripleVGA which has the number of the palette The higher the overall values of each of the three color

the video card into one of its possible video modes, and code for sending referenced by the BIOS. This is made up of tables, code for initializing pixels of colors to the screen encoded on it important information about the video card which is then looks to the Video Card installed in your machine. The video card has The palette is stored by the Video BIOS. When the computer starts up, it

What Can You Use Palettes For?

routines for displaying PCX graphics files. These graphics files were probably made with PC Paintbrush or some other graphics program (possibly even with the Graphics Workshop SavePCXVE subprogram). These programs all have the ability internally to change the palette of the Now that we have the code and knowledge for using palettes, what immediate uses do we have for it? In Graphics Workshop there are many

CHARLE MANAGER

■ B-2

S. P. L. B. in the overall r.Soha Vahre district and some solutions

To display these PCX files, use the native palette which was used to create it. Other uses discussed below are covered in the demonstration program DEMOPAL BAS. Another use is drawing video data to the screen and then recoloring it without having to image being displayed or created. redraw it to the screen. One could also use palettes for the creation of a three dimensional video game. The terrain in this demo can have complex images (the lines) drawn over them and it would appear that the terrain is being redrawn each time you want to move it. Obviously, redrawing the background image under these complex images would be very complicated and slow. Instead, the color of individual places on the screen is modified, allowing the program to spend valuable CPU time more efficiently. Palettes are good for displaying motion without actually using effort to redraw the screen.

6 4 Palent register Sett. De Colin f the palette registers mation for yourself.

Each of the colors follows one another in a succession which can make The Logo contained in the demo program makes use of a flowing palette. the image appear to be a solid color.

e ability to improve rs. You can specify en, and blue. This Traphics Workshop mber of the palette Blue color values, of the three color

color mode allows

mentioned above.

choose from, but PC and that's the ard which is then de for initializing

iputer starts up, il ne video card has code for sending

g palettes, what aphics files were raphics program

or there are many

These types of effects will make a presentation much more effective than is possible with just one color. Appendices

r e palette of the

IT SOFTWARE, AND

E subprogram!

Appendices ON SPRINGLING MANAGES is represented by a 0, w Many of the items in the simply as Yes or No indic APPENDIX C: THE GPDAT%() AI Description of the (with the package, but so such as allowing or ariables. System variables such as allowing variables. System variables are ther variables are popular work with the following such as a such which should be inclu GPDar%() array is an a of these are needed for of which the first 80 arc common statements has routines are font defir variables in your man variables which statement in Qu. (2), subroutine, Variable monitors, we've use GPDat%() array has maximum pixel reso routines, which we h fonts are those which values for using the (on the display, the co Professional GPDat sional, and since it's to use some of the he wondering why t It is a package for d GraphPak Profession We recognize that a Inside a subroutine The GPDat%() array

APPENDIX C:

THE GPDAT%() ARRAY FROM GRAPHPAK

It is a package for doing business charts and scientific graphs. You may We recognize that a lot of GraphPak Professional owners are going to want GraphPak Professional is another product produced by Crescent Software. be wondering why we talk about GraphPak Professional in this product. to use some of the Graphics Workshop routines with GraphPak Professional, and since it's nice to have some of the features of the GraphPak Professional GPDat%() array in any program having to deal with graphics monitors, we've used it in Graphics Workshop. GraphPak's Professional GPDat %() array has elements which hold the current screen mode set, the maximum pixel resolution of the display, the maximum number of colors on the display, the color values for pull-down and vertical menus, and vital values for using the GraphPak Professional fonts. GraphPak Professional fonts are those which are displayed using the DrawText and StepText routines, which we have also supplied in this product.

Description of the GPDat%() Array

Inside a subroutine in QuickBASIC, you don't have access to all the variables in your main program. Usually you'll have access only to those variables which you have passed to the subroutine. Using the COMMON The GPDat% () array was created to share information between routines. statement in QuickBASIC, you can reach variables from outside the Variables that we want to be able to access from inside various routines are font definitions, and the GPDat%() array. The requisite common statements have all been included in the COMMON.GW module GPDat%() array is an array of integers. It is dimensioned to 85 elements, of which the first 80 are used by GraphPak Professional routines. Not all of these are needed for the GraphPak Professional font routines included with this package, but some elements which are needed have significant GPDat%() array is broken up into two sections, system variables and user variables. System variables hold important system parameters, like screen User variables are used to modify the routines like DrawText and which should be included at the top of each of your modules. uses, such as allowing you to specify boldfacing, italicizing, etc. StepText work with the font information given to them.

Many of the items in the GPDat%() array are integers, but some are used simply as Yes or No indications, or boolean variables. A boolean variable is one that has two states, either TRUE or FALSE. In computers a FALSE is represented by a 0, whereas TRUE is commonly represented by -1.

Appendices

See the detail on the

Elements 14, 23, 24

-1, as it is more of a standard. BASIC considers any non-zero value to be TRUE, but try always to use

Elements Used By Graphics Workshop

_									_		_			acutation and	_				_		_	_				_			_	_		_
86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	59	57	50	49	48	47	44	43	36	35	34	33	31	24	23	14	ELEMENT
Draw an Outline Only	GW Overall Font Height	GW Vert Font Spacing	GW Horiz Font Spacing	GW Font Active	GW Fonts Available	Normal Screen Color	InActive Menu Color	Active Menu Color	InActive Item Color	Active Item Color	Pull-down Box Color	Highlight Bar Color	Mouse Active	Graphics Storage Segment	Character Height	Temporary Text Spacing	Background Screen Color	Maximum Colors	Screen Aspect Ratio	Vertical Text Spacing	Horizontal Text Spacing	Video Monitor Height	Video Monitor Width	Font Vertical Spacing	Maximum Font Width	Current Font Number	Maximum Fonts Available	Monitor Type	Italicize Text	Boldface Text	Text Shadow Color	VARIABLE DESCRIPTION
Boolean (-1 or 0)	Integer	Integer	Integer	Integer	Integer	Integer (GW color)	Integer (GW color)	Integer (GW color)	Integer (GW color)	Integer (GW color)	Integer (GW color)	Integer (GW Color)	Boolean (-1 or 0)	Integer (> &HA800)	Integer (8, 14, 16)	Integer (-15 to 15)	Integer (0 to 15)	Available	Integer (0 - 1000)	Integer	Integer	Integer (200, 350, or 480)	Integer (320, 640, or 720)	Integer	Integer	Integer (< = Fonts Avail)	Integer	Integer (0-9)	Integer (45 to 135)	Boolean (-1 or 0)	Integer (0-15)	DATA (RANGE)

Note:

Variables without ranges can have any range available to an integer variable. A GW color refers to colors designed to be displayed with the GPrint0VE routine. The formula for a GW (Graphics Workshop) color

GPDat%(23): Boldface Data Type:

This element is automate

Note:

Text Shadowing can concentrations. This elec-

Uses:

maximum number of c

this variable will be co mode (e.g. on CGA 4. higher values will disp Must be between 0 and

Possible Values

Data Type:

Integer

GPDat%(14): Text S

Graphics Workshop Detailed Listing Of

of these structures.

menus and message

font system. Elements 81 through Elements 74 through

which give vital info Elements 31, 43, 4 GraphPak Professio

Appendices

ON SHAMAIN MANAGE

GWcolor = Foreground + 256 * Background

See the detail on the following pages for more information.

Elements 14, 23, 24, 33, 34, 35, 36, 47, 48 and 59 all have effects on the GraphPak Professional proportional font system.

Elements 31, 43, 44, 49, 50, 57, 71, 72 and 73 are all system variables which give vital information about the screen mode in use and if a mouse is present.

5 to 135

Elements 74 through 80 all are used by the graphics pull-down, vertical menus and message box routines to pass the colors of the individual parts of these structures.

> 20, 640, or 720) 20, 350, or 480)

991

92

Elements 81 through 86 all have effects on the Graphics Workshop Vector font system.

Detailed Listing Of Elements Used In Graphics Workshop

GPDat%(14): Text Shadow Color

■ Data Type:

010

Integer

■ Possible Values:

Must be between 0 and 15. On some screen modes the range is less, but higher values will display as the highest available color for that screen mode (e.g. on CGA 4-color mode the range is 0 to 3. A value of 14 in this variable will be considered as a 3). GPDat%() element 50 has the maximum number of colors for the current screen mode.

■ Uses:

Text Shadowing can create interesting displays for major titles and This element affects only the GraphPak fonts.

■ Note:

0

This element is automatically initialized to zero by QuickBASIC.

GPDat%(23): Boldface Text

Data Type:

e to an integer

Boolean

Appendices

Possible Values:

-1 = Boldface the text

0 = Normal text

Uses:

affects only the GraphPak fonts. Boldfacing adds emphasis to points of interest on a graph. This element

Note:

within GETVIDEO. This variable is initi the correct screen m

the appropriate posi-This variable is used

Note:

This element is automatically initialized to zero by QuickBASIC

GPDat%(24): Italicize Text

Data Type:

Possible Values:

Must be at least I an

Data Type: GPDat%(33): Max

Integer

I Uses:

When the Current Fo

variable is used to en

OPPONT OW will o

Integer

Possible Values:

This variable represents the angle for italicizing. Normal text uses angle of 90 degrees. To italicize, use an angle of about 60 degrees. this variable is 45 to 135 degrees. obtain backwards italics use an angle above 90. A reasonable range for Normal text uses an

Uses:

affects only the GraphPak fonts. Italicizing helps to emphasize points of interest on a graph. This element

Note:

GETVIDEO.BAS initializes this variable to 90.

Note:

affects only the

number of fonts

of Max GPFonts % if th

GPDat%(31): Monitor Type Used

Data Type:

Integer

Possible Values:

- 0 = Monitor Unknown
- EGA with Mono Monitor
- Hercules Graphics Adaptor
- Monochrome (not capable of graphics)
- CGA Graphics 4 colors
- EGA Graphics 16 colors

· Uses:

(See GPDat%(33)) Must be at least I and c

usly the Graph Pak font y Allows for different for usge of this leature can p

Possible Values:

Data Type: GPDat%(34): Curren

- CGA Graphics 2 colors
- Mono EGA Graphics
- VGA Graphics 16 colors

Appendices

VGA Graphics 256 colors

CHECENT SUFTWARE INC.

This variable is used by most of the Graphics Workshop routines to create the appropriate positioning and scaling. It is also used by SetVideo to set the correct screen mode.

This element

unckB4SIC

This variable is initialized by the function MultMonitor, which is called within GETVIDEO.BAS.

GPDat%(33): Maximum Fonts Available

Data Type:

Integer

Possible Values:

Must be at least 1 and can be as large as string memory will allow.

omal text uses an isonable range for

ut 60 degrees. 7

When the Current Font Number is set using the routine SetGPFont, this variable is used to ensure that the font actually exists. The include file GPFONT.GW will dimension the Fonts() array to 95 elements by GPDat %(33) elements in size. If you want to have more than one GraphPak Professional font, set the variable MaxGPFonts % to the desired This element number of fonts prior to the include file GPFONT.GW. affects only the GraphPak font system.

ph. This element

GETVIDEO. BAS initializes this variable to 1. It will be reset to the value of MaxGPFonts % if the variable/constant contains a non-zero value.

GPDat%(34): Current Font Number

Data Type:

Integer

Possible Values:

Must be at least 1 and can be as large as the maximum number of fonts.

(See GPDat%(33))

Proper usage of this feature can produce spectacular effects. This element affects Allows for different fonts to be displayed on the same screen. only the GraphPak font system.

Appendices

■ C-5

Note:

GETVIDEO.BAS initializes this variable to 1.

GPDat%(35): Maximum Font Width

Data Type:

Integer

Possible Values:

measuring the width of the capital letter 'W'. In most fonts this will be the widest letter. Any value between 0 and 16 is possible. The value is determined by

Uses:

Allows StepText to display a centered label, while still using the proportional font system. This element affects only the GraphPak font system.

■ Possible Values:

Possible values are 20

Data Type: GPDat%(44): Video Note:

This variable is set by

actually used.

Uses:

outside the screen's Many routines use the Pussible values are 3

_ Uses:

Many routines use th

outside the screen's h

This variable is set by the routine SetGPFont.

GPDat%(36): Font Vertical Spacing

Data Type:

Integer

Possible Values:

value will typically be between 8 and 16. Font's Maximum Height is contained in the FontHeight%() array. Font's Maximum Height as determined by the LoadFont routine. This variable is set to the combination of GPDat%() element 48 and the

to another. This element affects only the GraphPak font system. Many of the routines use this variable to determine spacing from one line

Possible Values:

Any value from -15 to

■ Data Type:

GPDat%(47): Horizo

Note:

This variable is:

Note:

This variable is set by the routine SetGPFont

GPDat%(43): Video Monitor Width

Data Type:

Integer

- Note:

character. This element letters printed on the sc This variable sets the nu

This is set by the SetO VIDEOBAS IN 2 pixels

C-6

Appendices

WAXER SHAMPS INC.

Possible values are 320, 640 or 720, depending on the screen being used.

Many routines use this value to determine centering, and to avoid writing outside the screen's boundaries.

ie is determined by st fons this will be

This variable is set by the routine Set Video, depending on the screen mode

GPDat%(44): Video Monitor Height

Data Type:

Il using the proporphPak foot system.

Integer

Possible Values:

Possible values are 200, 350 or 480, depending on the screen being used.

Many routines use this value to determine centering, and to avoid writing outside the screen's boundaries.

This variable is set by the routine Set Video, depending on the screen mode actually used.

GPDat%(47): Horizontal Text Spacing

ont routine. The 15% array. This

ement 48 and the

Data Type:

Integer

Possible Values:

ing from one line

system.

Any value from -15 to 15 can be used.

This variable sets the number of pixels to appear as separators of any two letters printed on the screen. It is also the natural width for the space character. This element affects only the GraphPak font system.

This is set by the SetGPSpacing routine, but is initialized in GET-VIDEO BAS to 2 pixels.

Appendices

■ C-7

GPDat%(48): Vertical Text Spacing

Data Type:

I Uses:

the routine SetVides Possible values are

have to work with. This variable allows

Possible Values:

Any value from -15 to 15 can be used.

Uses:

This variable sets the number of pixels to move beyond the descender of any of the lower case letters like 'g', 'y', and 'j', before writing the next line of text. This element affects only the GraphPak font system.

Note:

This is set by the SetGPSpacing routine, but is initialized in GET-VIDEO.BAS to 2 pixels.

■ Possible Values:

Possible colors are 0

Data Type: GPDat%(57): Back

This value is set in t

Uses:

The routine GPaintBo

oxlor of the screen

GPDat%(49): Screen Aspect Ratio

Data Type:

Possible Values:

equation: Any value between 0 and 1000. This value is determined by the following

Uses:

This is used for drawing text at angles and drawing circles. The purpose of this value is to allow circles to appear perfectly circular, no matter what the screen resolution. It can also be used to adjust results of the SIN images. function when used for displaying rotated three-dimensional graphics

Possible Values:

Pixel offset from -15 to

the text offset, all text routines Data Type:

GPDat%(59): Tem

Note:

actually used. This value is set in the routine SetVideo, depending on the screen mode

GPDat%(50): Maximum Colors Available

Data Type:

Integer

C - 8

Appendices

CHI THUMPING LINCOLD

This variable is initialize

element affects only the element 47. It is good p GPDat%() System elem

after each character. Yo One use is in underlin

finished using it.

4, 16, or 256 depending on the screen activated in Possible values are 2, the routine SetVideo.

This variable allows the internal routines to know how many colors they have to work with.

Note:

nd the descender of in e writing the next

f mt System.

This value is set in the routine SetVideo.

GPDat%(57): Background Screen Color

Data Type:

nitialized in GET.

Possible Values:

Possible colors are 0 - 15.

The routine GPaintBox uses this variable to determine the background color of the screen.

GPDat%(59): Temporary Text Spacing

ad by the following

Data Type:

Integer

Possible Values:

Pixel offset from -15 to 15 for the base font size. To achieve a true pixel offset, all text routines automatically multiply this number by the size of

Uses:

esults of the SIN les. The purpose

ensional graphics ar, no matter what

after each character. You can set this variable to counteract the effects of element 47. It is good practice not to change the System elements.) This element affects only the GraphPak font system. One use is in underlining text, since all proportional fonts have spaces GPDat%() System element 47. (Use this element rather than changing

the screen mode

This variable is initialized to 0. Reset this variable to 0 when you have finished using it.

Appendices

■ C-9

I Mouse driver is

O.Mouse not availa

I Uses:

The routines PullD

whether or not to ex

Note:

Init/Youse routine. This variable is set I

GPDat%(71): Character Height

Data Type:

Integer

Possible Values:

For the CGA displays the height of the character font is 8 pixels. For EGA displays the height is 14 pixels. For VGA displays the height is 16 pixels.

ment of graphics elements, and for conversion between coordinate sysmixed system. The PullDownG and VertMenuG systems use this information for place-This value can also be used for placement of text when using the

GPDat%(74): High

■ Data Type:

Note:

actually used. This variable is set by the routine SetVideo, depending on the screen mode

■ Possible Values:

Possible values includ

nutine. The formula

GPDat%(72): Graphics Storage Segment

Data Type:

Integer

Possible Values:

or EGA video memory, it is going to be some value above &HA800. The video page. video page, if there is enough memory to have a complete second VGA The segment &HAA00 will be the starting location for the second segment &HA800 is the starting location for the second EGA video page Since this variable is used to store the next possible location to save VGA

Uses:

value to know where they can save their next information. The routines PullDownG, VertMenuG, and MsgBoxG use this segment

Note:

MsgBoxG. This variable is set inside the routines PullDownG, VertMenuG, and

Possible Values:

Possible values include o

routine. The formula to

GPDat%(75): Pull-do Data Type:

Note:

This variable is set ins

■ Uses:

The routines

GPDat%(73): Mouse Active

Data Type:

Boolean

Appendices

DESCRIPTION PARTIES

· Uses:

icolor

The course Pull Awar determine colors on the

-1 Mouse driver is loaded and mouse initialized

0 Mouse not available

8 Ives For EGA

he at is 16 pixels.

The routines PullDownG and VertMenuG use this value to determine whether or not to execute mouse specific code.

Note:

imation for place.

en coordinate sysext when using the

This variable is set inside the include file GETVIDEO.BAS by calling the InitMouse routine.

GPDat%(74): Highlight Bar Color

Data Type:

Integer

on the screen mode

Possible Values:

Possible values include color values which are designed for the GPrintOVE The formula for combining a foreground color and a background routine. color is

The routines PullDownG, VertMenuG, and MsgBoxG use this color to determine colors on the screen under the right circumstances.

tion to save VGA & &HA800. The EGA video page. the second VGA ete second VGA

This variable is set inside the include file GETVIDEO.BAS.

GPDat%(75): Pull-down Box Color

Data Type:

Integer

ise this segment

Possible Values:

Possible values include color values which are designed for the GPrintOVE The formula for combining a foreground color and a background color is

ert.MenuG, and

Uses:

The routines PullDownG, VertMenuG, and MsgBoxG use this color to determine colors on the screen under the right circumstances.

SOFTWARE, INC.

Appendices

Assible values inclu

Note:

This variable is set inside the include file GETVIDEO.BAS

GPDat%(76): Active Item Color

Data Type:

Integer

Possible Values:

color is routine. Possible values include color values which are designed for the GPrintOVE The formula for combining a foreground color and a background

Note:

The routines PullDo

determine colors on

This variable is set li

Uses:

determine colors on the screen under the right circumstances. The routines PullDownG, VertMenuG, and MsgBoxG use this color to

■ Possible Values:

routine. The formula Possible values includ ■ Data Type: GPDat%(79): InAct

Note:

This variable is set inside the include file GETVIDEO.BAS

GPDat%(77): InActive Item Color

Data Type:

Integer

Possible Values:

color is routine. Possible values include color values which are designed for the GPrint0VE The formula for combining a foreground color and a background

determine colors on the screen under the right circumstances. The routines PullDownG, VertMenuG, and MsgBoxG use this color to

Note:

This variable is set inside the include file GETVIDEO.BAS

Possible Values:

Toutine. The formula fo Possible values include c

GPDat%(80): Norma ■ Data Type:

Note:

This variable is set ins

■ Uses:

The routines Pulls

determine colors of

GPDat%(78): Active Menu Color

Data Type:

Integer

Appendices

CHESCENT SUFFINALE DAC

on to stolog aumana The routies Pullbown

· Uses:

Possible values include color values which are designed for the GPrintOVE routine. The formula for combining a foreground color and a background color is

GColor = Foreground + 256 * Background

■ Uses:

forthe Optintove

The routines PullDownG, VertMenuG, and MsgBoxG use this color to determine colors on the screen under the right circumstances.

Note:

This variable is set inside the include file GETVIDEO.BAS.

GPDat%(79): InActive Menu Color

■ Data Type:

use this color to

Alles.

Integer

■ Possible Values:

Possible values include color values which are designed for the GPrintOVE routine. The formula for combining a foreground color and a background color is

■ Uses:

The routines PullDownG, VertMenuG, and MsgBoxG use this color to determine colors on the screen under the right circumstances.

Note

This variable is set inside the include file GETVIDEO.BAS.

GPDat%(80): Normal Screen Color

Data Type: Integer

ise this color to

■ Possible Values:

Possible values include color values which are designed for the GPrintOVE The formula for combining a foreground color and a background routine. color is

l leoc.

The routines PullDownG, VertMenuG, and MsgBoxG use this color to determine colors on the screen under the right circumstances.

Appendices

SOFTWARE, INC.

This can be any non-

Uses:

affects only the Grap This variable sets the

Note:

pixels in GETVIDE This variable is set l

Note:

This variable is set inside the include file GETVIDEO.BAS

GPDat%(81): GW Fonts Available

Data Type:

Integer

Possible Values:

Depends on the number of fonts requested.

Uses:

you want to have more than one Graphics Workshop vector font, set the variable MaxGWFonts% to the number of fonts desired, prior to the SetGWFont, this variable is used to ensure that the font actually exists. If Workshop Vector fonts. include file GWFONT.GW. When the Graphics Workshop Font Active is set using the routine This element affects only the Graphics

Note:

This variable is set in the include file GWFONT.GW. The value will be 1, if no value has been placed in the variable MaxGWFonts %.

GPDat%(82): GW Font Active

■ Note:

This variable is set b

'y' and 'j'. This elem

line of text beyond th

I Uses:

This variable sets the

Possible Values:

This can be any non-I

GPDat%(84): GW

Data Type:

Integer

Data Type:

Integer

Possible Values:

GPDat % (81) Must be less than or equal to the number of fonts available.

Uses:

string. This variable tells the routines which of the fonts to use when drawing a This element affects only the Graphics Workshop Vector fonts.

Possible Values:

■ Data Type:

GPDat%(85): GW (

Note:

This variable is set by the routine SetGWFont

GPDat%(83): GW Horizontal Font Spacing

Data Type:

Integer

Appendices

ORIGENT STATISTICS

Note:

to another. This elemen Nany of the routines us

This variable is set b

Selfwspacing.

· Uses:

FontSize % () routine. The Graphic Workshop Font's Maxir This variable is set to the

This can be any non-negative number. A reasonable number is 5 (pixels).

This variable sets the spacing left and right between letters. This element affects only the Graphics Workshop Vector fonts.

This variable is set by the routine SetGWSpacing, but is initialized to 5 pixels in GETVIDEO BAS

GPDat%(84): GW Vertical Font Spacing

Data Type:

ired, prior to the

using the routine actually exists. If ector font, set the

Possible Values:

This can be any non-negative number. A reasonable number is 5 (pixels).

The value will be

This variable sets the number of pixels to move before writing the next 'y' and 'j'. This element affects only the Graphics Workshop Vector fonts. line of text beyond the descender of any of the lower case letters like 'g',

This variable is set by the routine SetGWSpacing, but is initialized to 5 pixels in GETVIDEO BAS.

GPDat%(85): GW Overall Font Height

Data Type:

Integer

Possible Values:

when drawing a

Vector fonts.

This variable is set to the sum of GPDat%() element 84 and the Graphics Workshop Font's Maximum Height as determined by the LoadOutlineFont The Graphics Workshop Font Height is stored in the array FontSize%(). routine.

Many of the routines use this variable to determine spacing from one line to another. This element affects only the Graphics Workshop Vector fonts.

This variable is set by the routine SetGWFont and by the routine SetGWSpacing.

DETWARE, DIC

GRAPHPAK PROF CONVERTING FR

APPENDIX D:

GPDat%(86): Draw an Outline Only

Data Type:

Boolean

- Possible Values:
- -1 Draw Outline Only
- 0 Fill Font In

Uses:

a different color than the rest of the font. This function allows you to create an outline font, or by drawing an outline font over a filled in font, it allows you to create a font with an outline of

Converting from C

One of the major diff

the background color foreground color is s the color values are Because QuickPak P. going to be using the transfer to make the tra

Many users from Q

Note:

This variable is initialized in GETVIDEO.BAS to 0.

Appendices

■ C-16

CRESCENT SOFTWARE, INC.

ON BRANLING LIBORE

efficient. The figure h

Rather than pack both for the entire screen is

supports different bac location of the screen i when in graphics mo of the colors at any or

Dackground odor, No

ground color. Ther

APPENDIX D: CONVERTING FROM QUICKPAK OR GRAPHPAK PROFESSIONAL

Many users from QuickPak Professional and GraphPak Professional are going to be using the features in Graphics Workshop. This Appendix is here to make the transition easier.

Converting from QuickPak Professional

t with an outline of

the color values are optimized for that mode. Briefly, in text mode the One of the major differences between the two packages is the use of color. Because QuickPak Professional routines are designed for the text modes, foreground color is stored in the lower 4 bits of a single color byte, and the background color in the higher 4 bits.



Graphics memory is very different, because there is no explicit background color. There are many colors, and it is impossible to decide which of the colors at any one location on the screen should be considered the background color. Note that BASIC treats color 0 as the background color when in graphics mode. If you modify the background color for any location of the screen using the COLOR statement, the background color for the entire screen is changed. Graphics Workshop uses a system which supports different background colors at different regions of the screen. Rather than pack both colors into a single byte, the Graphics Workshop Since an integer holds two bytes, this both faster and The figure below shows how these bytes are organized. uses two bytes. efficient.



OFTWARE BY

■ D - 1

to isolate the high byte to obtain the background color. bits is not required to form the background color. All that is necessary is It is also faster to manipulate colors stored this way, because rotation of

GraphPak Professi

changes will be not

apter 1 at the bes

with the following formula: In QuickPak Professional, a color for printing a text string is formulated

Colr% = Foreground% + Background% * 16

The Graphics Workshop formula is changed as follows:

to GraphPak Profes SIMPLE BAS. SIM that instead of i

Professional and It

Colr% = Foreground% + Background% * 256

sional. The following is a brief table comparing similar routines: the next step is to find equivalent routines for those in QuickPak Profes-Once the color values have been changed to work with the graphics modes,

QuickPak Professional	Graphics Workshop
QPrintRC	GPrint0VE
ScrnSaveRest	GMove2VE
PaintBox	GPaintBox

Graphics Workshop one that comes with to your current direc You should also cop

file that comes will You must also use the inaphPak's COMMi

combined foreground-background color value. Some of the QuickPak The routine GPrintOVE requires a cursor position, a text string, and a the POS(0) statements to find the current cursor location. to you to know where the cursor is located. You can use the CSRLIN and printing routines accept a row and column, while others do not. It is up

the visible page and higher video memory. One important difference between this routine and the QuickPak equivalent is that GMove2VE uses the mixed coordinate system, while QuickPak only deals with text screen The GMove2VE routine stores and retrieves graphics regions using both

either Landscape or Po parameters which per hanced, and hence has GraphPak Professi The routines by

there% has remaine

routine. single character coordinate on the screen. The one important use for The primary difference is that only one color will be changed by this be changed. In graphics mode it is possible for 16 colors to exist in a character location of the screen, it is simple to specify what colors should closely as possible. GPaintBox is to create a shadow effect in the pull-down and vertical menus. The GPaintBox routine emulates the PaintBox and MPaintBox routines as In addition, the GPaintBox routine uses the mixed coordinate First, since text mode has only two colors for any

Combining with GraphPak Professional

chapter 1 at the beginning of your programs. The one minor change is that instead of including GETVIDEO.BAS you should include GraphPak Professional is already a graphics mode program, so fewer You should use the "standard" code shown in SIMPLE BAS. SIMPLE BAS is the include file that came with GraphPak Professional and it contains more setup information which pertains only to GraphPak Professional. In your standard code, replace the line that changes will be necessary.

'\$INCLUDE: 'GETVIDEO.BAS'

the graphis modes, "QuickPak Profes-

lar routines.

'\$INCLUDE: 'SIMPLE.BAS'

You should also copy the GETVIDEO.BAS file from Graphics Workshop to your current directory. It performs many of the same functions as the one that comes with GraphPak Professional, but it handles more of the Graphics Workshop setup as well.

> text string, and a S do not. It is up of the QuickPak the CSRLIN and

file that comes with GraphPak. All of the variables contained in GraphPak's COMMON.BAS file are also contained in the COMMON.GW You must also use the COMMON. GW file instead of the COMMON. BAS

Herc-The routines HercThere% and ScrnDump which were contained in GraphPak Professional are also a part of Graphics Workshop. Herc-There% has remained unchanged, but ScrnDump has been greatly enhanced, and hence has been re-named to ScrnDump2. It has three added parameters which perform simple scaling, and tell the routine to print in either Landscape or Portrait mode.

> o colors for any ors to exist in a portant use for changed by this ixed coordinate

at colors should

vertical menus.

(Box routines 23

with text screen

Appendices STEWERT WASHINGER W.C. The Gercland-Vers of quested the physical countries of the countries of th APPENDIX E: IMPROVING PI) The first eight horizo stored in a single by and so forth. To calc You be streen you and the year of a walle by 8. In as multiple of 2 using b normal division. How taker than usual. By passing all loss contains, a cr accessing the accessing improve the speed this section. Plea The primary purp on BASIC's speed read from or writter reduces the number A cache is an area it to successfully us presented solely for The Graphics Wor itself to a variety of on BASIC which contribute which contribute which contribute haps most import haps most received another is and yet another is another is another is another is another in the contribute of the contribute in the contribute is a second contribute in the contribute in the contribute is a second contribute in the contribute in the contribute is a second contribute in the contribute in th with a disk cache,

APPENDIX E:

IMPROVING PIXEL ACCESS USING A CACHE BUFFER

The primary purpose of providing low-level graphics routines is to improve on BASIC's speed, as well as add capabilities. There are several factors which contribute to the slowness of manipulating graphics images. Perhaps most important is the need to access screen memory one pixel at a time. Another factor is the time required to pass and receive parameters, and yet another is the internal calculation to determine a pixel's address. Fortunately, EGA and VGA memory is organized in a manner that lends itself to a variety of speed-up techniques.

The Graphics Workshop uses a technique known as cache buffering to Please understand that the information contained here is presented solely for completeness, and you do not need to fully understand it to successfully use the Graphics Workshop. improve the speed of its low-level video routines, and it is described this section.

A cache is an area of memory which is used to retain information being with a disk cache, which serves a similar purpose. In that case, a cache read from or written to other, slower memory. You may already be familiar By passing all data through the cache and remembering which data it contains, a cache routine can return the data from the cache instead of In the Graphics Workshop, a cache is used to avoid reading the relatively slow EGA and VGA memory when the same pixel or nearby values are accessed more reduces the number of times the disk drive must be physically accessed. To appreciate how this cache is designed requires an underaccessing the slower memory or hardware device. standing of how EGA and VGA memory is organized. than once.

The first eight horizontal pixels at the upper-left corner of the screen are stored in a single byte at address 0. The next 8 pixels are at address 1, and so forth. To calculate the address of a particular pixel at location (X, Y) on the screen you simply multiply the Y value times 80, and divide the X value by 8. In assembly language a number can be divided by any multiple of 2 using bit-shifting operations, and this is must faster than a normal division. However, the multiplication is still necessary. If seven out of eight multiplications can be avoided, the pixels will be accessed faster than usual.

quested, the physical video memory address for that coordinate, and the The GetCacheVE% routine remembers the last (X, Y) coordinate recoordinates match the same physical screen address, GetCacheVE% data at that address when the request was made. Then, if future (X,

or access the slower video memory. retrieves the data from its cache, without having to calculate an address

purpose:

sparified (X, Y) o GetCacheVE% is

operate more quit

Syntax:

Where:

EGA and VGA ser V% will receive a XPos% and YPos will know that its data is not current. CircleVE, all reset a flag within in the cache routine, so GetCacheVE% Since it is possible to circumvent the cache routine, the data in the cache buffer could become obsolete. This can happen when a non-Graphics Workshop routine writes directly to screen memory without going through The routines DrawPointVE, DrawPointVEOpts, LineVE, and

using GetCacheVE%. be called after using the BASIC LINE or CIRCLE command, but before let you reset the cache manually if necessary. ResetCache would therefore Workshop routines for all drawing, the ResetCache routine is provided to or even clearing the flag. our cache, they will change the screen but without updating the cache data routine such as LINE or DRAW. Since these statements are not aware of Another way that the cache data could become invalid is by using a BASIC Although we assume you will use Graphics

the cache buffer. This technique is shown in the comments that accompany routines, ReDrawVE does not require a pixel location. Rather, it simply We have also included the ReDrawVE routine, which lets you redraw the last pixel that was plotted. Unlike the other Graphics Workshop plotting calling ReDrawVE GetCacheVE% to specify the most recently accessed location before ReDrawVE. redraws the most recently accessed pixel, based on the data currently in Since it takes advantage of the cache, you must call

■ See Also:

GetCacheVE% func specify the pixel to

whose value was ju

A special routine, R

GetCacheVI

The routine

or modules which speed. Including th All parameters for I Comments:

DrawByteVE, Draw

OBSTANTANT MANAGER

GetCacheVE%

September 1 Brough & Street so GerCaute/Eg

T & S. A. Graphy

function contained in GW.LIB Assembler

Purpose:

GetCacheVE% is similar to GetPointVE%, except it uses a cache to operate more quickly. GetCacheVE% returns the color of the pixel at a specified (X, Y) coordinate.

by using a BASIC

is are not aware of ing the cache data will use Graphics fine is provided to e would therefore mand, but before

V% = GetCacheVE%(BYVAL XPos%, BYVAL YPos%)

Where:

XPos% and YPos% specify the (X, Y) coordinate.

V% will receive a color value between 0 and 15 for the high-resolution EGA and VGA screen modes.

Comments:

S you redraw the

data currently in Orkshop plotting Rather, it simply

you must call that accompany

All parameters for this routine are passed by value to provide the maximum speed. Including the file GWDECL. BAS at the beginning of all programs or modules which use this routine will insure proper operation. The routines DrawByteVE and DrawPointVE reset the cache so GetCacheVE% will return the correct value. If you use any other routine which modifies the pixels on the screen, use ResetCache before calling GetCacheVE% again.

whose value was just determined. ReDrawVE does not require you to specify the pixel to change as it assumes the last position specified by the A special routine, ReDrawVE, can be used to change the color of the point GetCacheVE% function.

See Also:

DrawByteVE, DrawPointVE, GetPointVE%, ReDrawVE, ResetCache

OPTHURE, INC.

ReDrawVE

subroutine contained in GW.LIB Assembler

Purpose:

ResetCache is us

GetCache VE% fi use a non-Graf

ensure that the G

Purpose:

GetCacheVE% routine. It utilizes GetCacheVE's cache to remember the last location accessed on the screen. ReDrawVE changes the color of the last point returned by the

Syntax:

CALL ReDrawVE(BYVAL PointColor%)

Syntax:

CALL ResetCache

Where:

PointColor% is a color between 0 and 15.

Comments:

or modules which use this routine will insure proper operation. The parameter for this routine is passed by value to provide the maximum speed. Including the file GWDECL.BAS at the beginning of all programs

eliminating extra parameters and the need to re-calculate the screen black to red within a specified region use: position of the point. For example, to change all occurrences of the color the color of any region of the screen with great speed. It does this by This routine, in conjunction with the GetCacheVE% function, can modify The function GetCacheVE% must be called prior to using this routine

NEXT FOR s% = 100 TO 150 NEXT FOR t% = 100 TO 150 V% = GetCacheVE%(s%,t%) IF V% = 0 THEN V% = 4 CALL ReDrawVE(V%)

> DrawPointVE, Dra See Also: reset the cache

for compatit

proper value.

statement, then a eache, and does n

is because the PSE return the original GetCache VE% 10 PSET statement If you make a call Comments:

See Also:

GetCacheVE%, DrawPointVE

Appendices

ONL'S BRANCHES LANGES

ResetCache

B) C) IIB

Tellined by the e w temember the

Assembler subroutine contained in GW.LIB

Purpose:

use a non-Graphics Workshop routine between two calls to the GetCacheVE% function, then you must call ResetCache between them to ensure that the GetCacheVE% cache is up-to-date. ResetCache is used to refresh the cache used by GetCacheVE%. If you

Syntax:

CALL ResetCache

Comments:

ide the maximum g of all programs

Tation.

ing this routine. tion, can modify It does this by late the screen nes of the oxlor

If you make a call to the GetCacheVE% function, and then use the BASIC PSET statement to write over that pixel, and once again call the GetCacheVE% routine to ask it what the color is, GetCacheVE% will cache, and does not update it. If you call ResetCache after the PSET statement, then a following GetCacheVE% statement will return the is because the PSET statement has no connection with the GetCacheVE% return the original value and not the value set by the PSET statement. proper value.

It is unlikely that you will need to call the ResetCache routine, but it exists for compatibility with BASIC graphics statements. A call to the DrawPointVE, DrawByteVE, LineVE, or CircleVE routines will also reset the cache.

See Also:

DrawPointVE, DrawByteVE, GetCacheVE%, LineVE, CircleVE



